1 Nexsis Product Overview

1.2 Chip Block Diagram

1.3 System Block Diagram

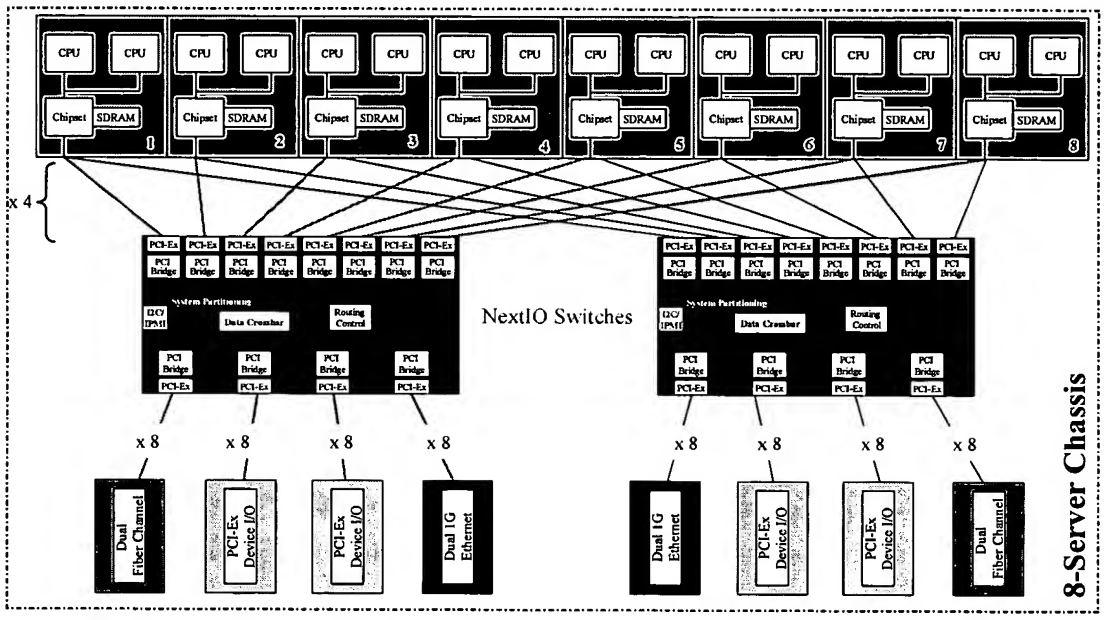


Figure 1.3-1: 8-Server Chassis Example

1.4.2 MAC Features

- PCI Express V1.0a compliant
- Can be configured to operate as a Shared Port or a Non-Shared Port.
- Layered architecture with Configuration Block, Presentation Layer, Transaction Layer, Data Link Layer, and Logical Physical Layer.
- Supports x1, x2, x4, and x8 links
- Supports up to 16 OS Domains and up to 8 Virtual Channels with a maximum of 16 different OSD/VC combinations.
- Provides a 64-bit data path at 250MHz
- Supports configurable maximum packet sizes in the range of 128B to 1KB
- Performs PHY level link negotiations

- The MAC is divided into 8 sub-modules, where each sub-module consists of 8 physical lanes that can be negotiated to operate as 1 x8 link, or 2 links of x1, x2, or x4 lane widths
- 64KB of Receive Data Buffer and 6144 Bytes of Header Buffer (256 outstanding transactions per port)
- Supports cannibalization of Receive Data Buffers and Receive Header Buffers. An 8x link can use all 128KB of Receive Data Buffer and all 512 Header Credits.
- 8-bit Single Error Correction and Double Error Detection (SECDED) on Rx Data & Header Buffers
- Maintains Transaction Ordering within each egress port per OSD/VC
- Add Power States supported
- Add latency statements (mention 'fast path' feature)

1.4.3 Switch Core Features

The switch core is responsible for forwarding transaction packet data from the 16 RX MAC interfaces over to the 16 TX MAC interfaces. This will be implemented as a data crossbar and a transaction scheduler. The high level functions of the switch core are:

- Implements a transaction scheduler that allows for programmable fairness at the different arbitration levels (per input OSD/VC per output port per VC) that transfers one transaction per clock from a source to a destination
- Provides virtual output queue (VOQ) structure so that each input feeds its own virtual switch from a software configuration viewpoint
- Implements a data crossbar that efficiently moves the packet data from the 16 RX MACs over to the 16 TX MACs
- Provides an interface to the switch management logic for transactions that terminate in our device
- Lookup interface for PCI bridge routing tables (support for address routing, ID-based bus routing, and implicit routing)

2 Nexsis Macro Architecture

2.1 D tailed Block Diagram

2.1.1 Chip Level Block Diagram

2.1.2 Block Descriptions

2.2 System View of Chip

2.2.1 PCI-Bus enumeration - Prashantha

2.2.2 Transaction Routing

According to the PCI Express Base Spec (1a), there are three ways to route a packet: address, ID, or implicit. The type of routing used is dependent on the Type field of the header (and the routing sub-field, r[2:0], of the Type field for message transactions). Each routing type will be covered in the next few sections.

In PCI Express, each switch is logically a set of virtual P2P bridges connected as shown below.

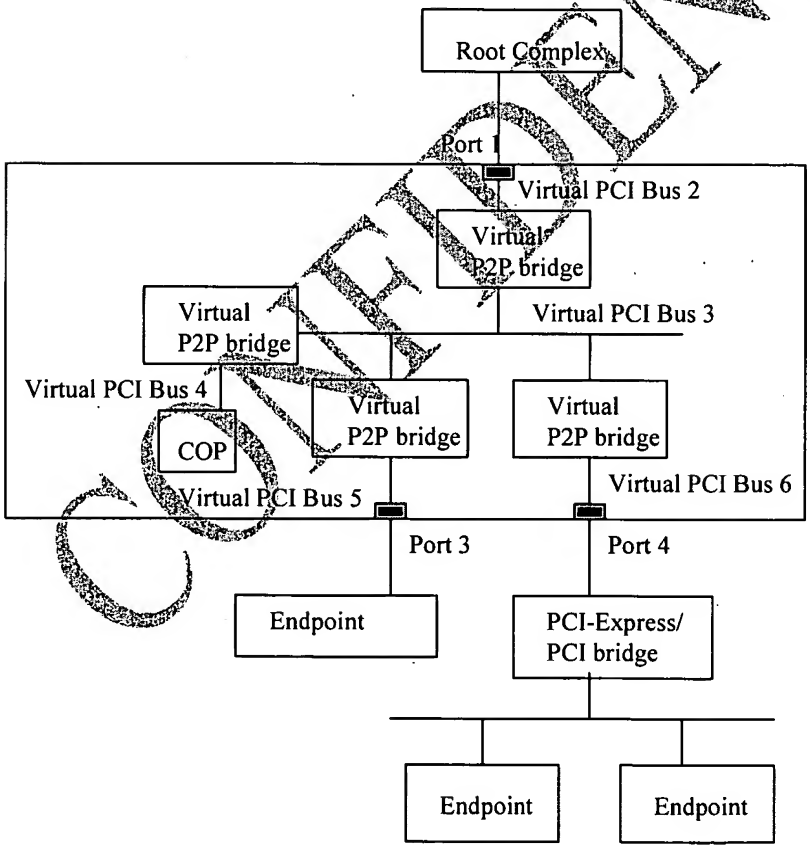


Figure 2.2-1:

This topology is a simplified version of our switch since it only shows 3 ports instead of 16, but it illustrates the idea by having 4 virtual P2P bridges, and 5 PCI buses, numbered 2 through 6. This topology will be used to show examples of how our routing will work and also shows where our on-chip device, the Common On-chip Processor (COP), will be located in the topology once discovery is completed by the root complex.

2.2.3 P2P bridges

A P2P bridge forwards memory and I/O transfers from one PCI bus over to the other side of the bridge where another PCI bus resides. Each P2P bridge has a 256B header register space that is accessible by the system and is used to set up all the parameters for the bridge. The following diagram shows the bridge header (Also called a Type 1 header) with the fields highlighted that apply to transaction routing.

Device ID		Vendor ID	
Status		Command	
Class Code			Revision ID
BIST	Header Type	Primary Latency Timer	Cache Line Size
Base Address Register 0			
Base Address Register 1			
Secondary Latency Timer	Subordinate Bus Number	Secondary Bus Number	Primary Bus Number
Secondary Status		I/O Limit	I/O Base
Memory Limit		Memory Base	
Prefetchable Memory Limit		Prefetchable Memory Base	
Prefetchable Base Upper 32 Bits			
Prefetchable Limit Upper 32 Bits			
I/O Limit Upper 16 Bits		I/O Base Upper 16 Bits	
Reserved			Capability Pointer
Expansion ROM Base Address			
Bridge Control		Interrupt Pin	Interrupt Line

Figure 2.2-2:

All of these registers are documented in the P2P Bridge spec (pp. 26-54), with some changes made in PCI Express (base spec 1a, section 7.5.3, pp. 327-330). The addressing starts at the top right of the table as register address 0x0 (lower 8 bits of Vendor ID) and continues to the bottom left as register 0x3F (upper 8 bits of Bridge Control).

2.2.4 Address routing

Address routing is used for memory (32-bit and 64-bit) and I/O transactions that must pass through our switch. Each will be discussed in detail in the next few sections.

The header fields important to address routing are shown in the next two diagrams.

+0	+1	+2	+3

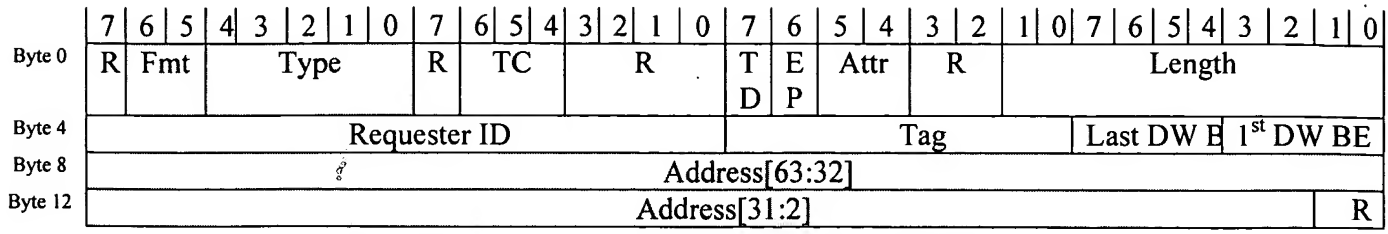


Figure 2.2-3: Header format for 64-bit Transactions (prefetchable memory)

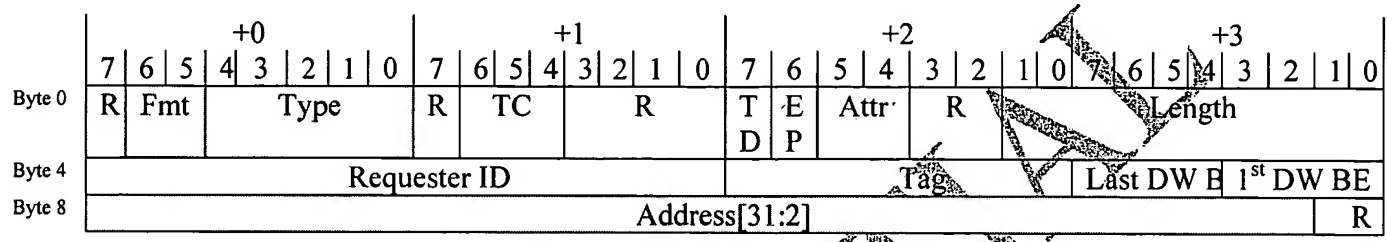


Figure 2.2-4: Header format for 32-bit Transactions (memory and I/O)

2.2.4.1.1 Memory transactions

In P2P bridges, there are 2 different address ranges that can be defined, the 32-bit memory range (which is required) and the 64-bit memory range (which is optional). Our switch will implement both ranges, so the bridge header registers for both memory transaction types will need to be set up by the system. The memory TLP in PCI express does not specify which of the two ranges a transaction will fit into (the SAC and DAC address cycles don't exist like they do in PCI), so both the memory range and the prefetchable memory range must be compared against for a 32-bit transaction since the prefetchable range can be below 4GB. A memory transaction is defined as follows:

TLP Type	Fmt[1:0]	Type[4:0]	Description
MRd	00 01	0 0000	Memory read
MRdLk	00 01	0 0001	Memory read-locked
MWr	10 11	0 0000	Memory write

Table 2.2-1:

The LSB in in the Fmt[1:0] field each of the three memory request types is high if a 64-bit address is present and low for a 32-bit address.

Each memory request (for decode purposes of the bridge) is addressing 1 MB of memory, so the lower 20 bits are assumed to be zero to match the memory space to the base/limit range. Note that any memory transaction that is addressing less than 4 GB is always a 32-bit transaction. Only transactions above 4 GB may use 64-bit address mode TLP and

are defined as prefetchable transactions that use the prefetchable base/limit registers. Following is a diagram that shows one way the system could provision the two memory address ranges.

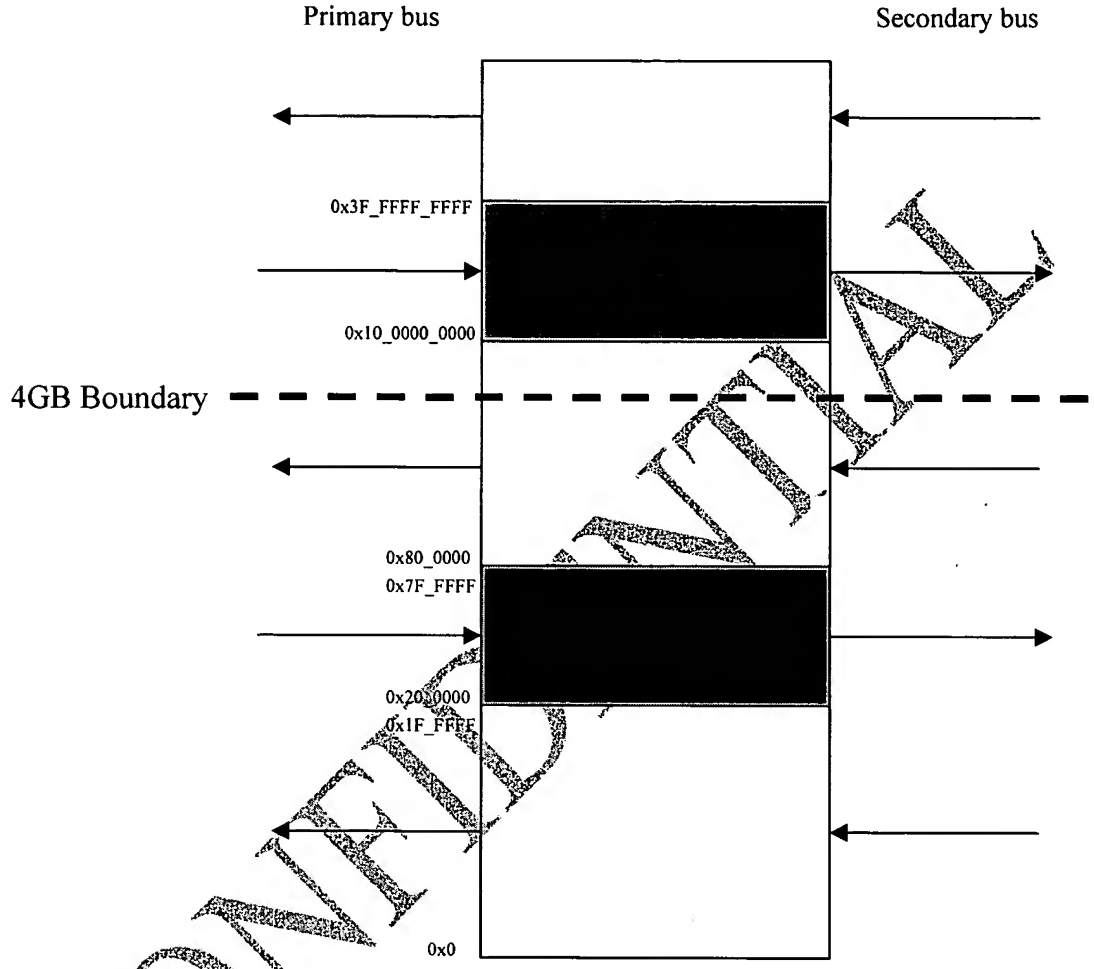


Figure 2.2-5:

In this diagram, the address 0x40_0000 going from north to south would still get through. Also, the address 0x20_0000_0000 would go through from north to south. The address 0xFFFF_FFFF_FFFF_FFFF from south to north would also pass across the bridge in this configuration.

Bottom line is that for 32-bit memory transactions, both the standard memory range, the prefetchable range, and the memory-mapped BAR space must be examined to process a transaction. For 64-bit transactions, only the prefetchable range must be examined.

2.2.4.1.2 I/O transactions

I/O transactions are limited to a 32-bit space, with a base and a limit being specified within the P2P bridge header just like memory transactions. The following fields are used to define the I/O TLP:

TLP Type	Fmt[1:0]	Type[4:0]	Description
IORd	00	0 0010	I/O read request
IOWr	10	0 0010	I/O write request

Table 2.2-2:

This type of access is 4kB-aligned (lower 12 bits are assumed to be zero for matching the range), so only the upper 20 bits of an I/O transfer are compared for a match in the address range. For transactions traveling downstream, the I/O address must match within the I/O range, and the transaction will be forwarded downstream. For transactions travelling upstream, the address must match outside the range, and the transaction will be forwarded upstream. The following diagram shows how a sample I/O address base/limit pair can be set up, using base of 0x20_0000 and a limit of 0x7F_FFFF.

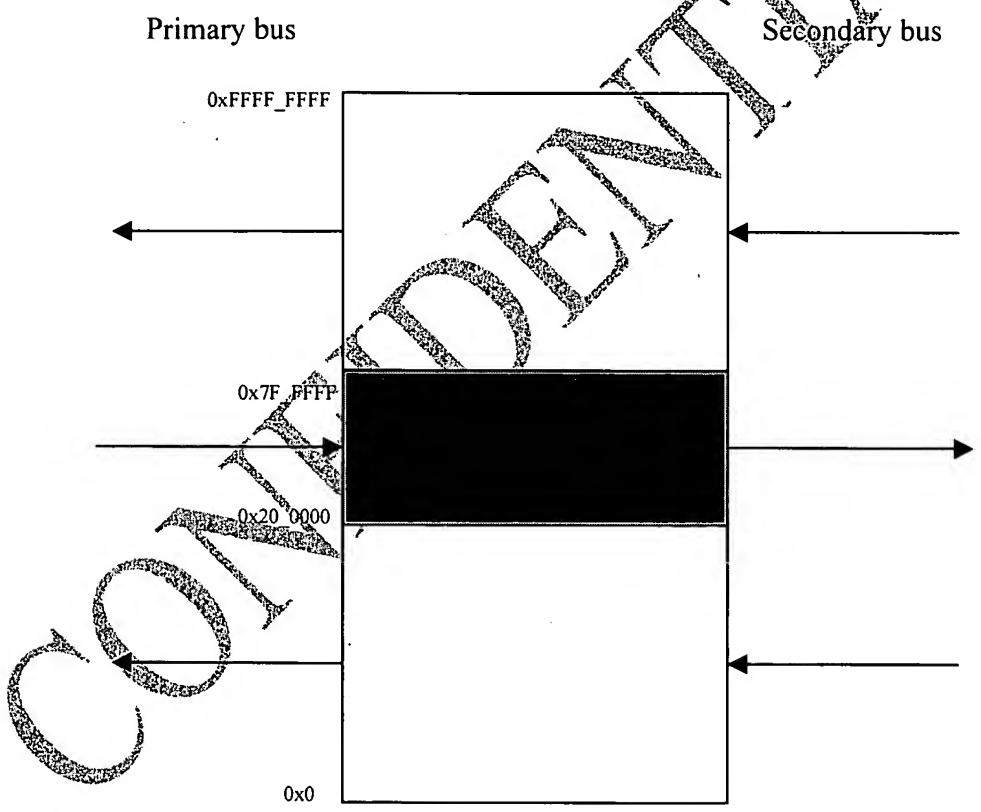


Figure 2.2-6:

For example, using this configuration, a transaction of 0x40_0000 is presented on the bridge's primary bus interface. Since it matches within the range programmed in the I/O base/limit registers, the transaction is forwarded downstream to the bridge's secondary bus. Next, a transaction for address 0x0 is presented on the bridge's secondary bus. Since this is NOT within the base/limit range, this transaction is forwarded up to the bridge's primary bus. In order to disable this range, the limit must be programmed to a

smaller number than the base. This means all downstream transactions will be rejected, and all upstream transactions will be allowed through. In the event of an error (an address of 0x10_0000 is presented at the primary bus for example), the transaction is discarded and an Unsupported Request (UR) message must be sent back out the port.

2.2.4.1.3 Peer to Peer support

Upstream traffic in a bridge is only allowed to cross the bridge if it's outside the {base,limit} range as described above in the examples. This would mean that if a downstream port initiates a read or write request, the address would need to fall outside the ranges of that port's P2P bridge.

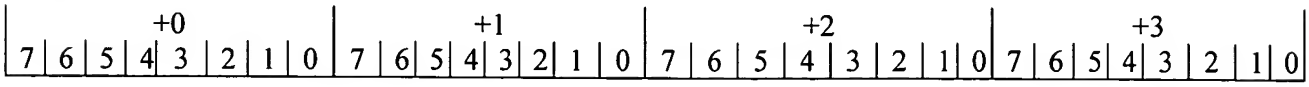
Assuming it's outside the range, the address is then compared with the root's address range and the other endpoint's address ranges. For the root's range, the address must again be outside the {base,limit} tuple. If it is, the transaction is forwarded up to the root. If the address is within the {base,limit} range, the other endpoint's ranges are checked for downstream matches. If a match is found, the transaction is forwarded to the other endpoint. If no match is found, the bridge issues a UR packet back to the endpoint.

The same mechanism will be used for downstream peer-to-peer support as well. The Root will be compared first since it is the likely recipient, but if no match is found the other downstream ports will be checked to see if the transaction should be routed there instead.

So, our architecture will be able to support IPC (inter-processor communication) and downstream peer-to-peer if the EEPROM provisions our switch such that two roots can appear on the same hierarchy. Our address_lookup_module logic does not preclude this from happening. We'll have a bit defined per base/limit pair that allows for a very flexible mapping. This bit will define each port as either upstream or downstream for each port for each OSD. This allows the address routing logic to map addresses into the ranges and allow for upstream peer-to-peer and downstream peer-to-peer in the same architecture. This is configurable by the EEPROM as to which ports "appear" on each OSD during device discovery. Note that only the trusted software entity (either a driver running on one OSD or the I2C interface software) is allowed to modify the routing bits, so OSDs will not be able to make other OSDs "appear" on their PCI hierarchy by accident. Any peer to peer configuration is application specific on how it handles the address mapping.

2.2.5 ID routing

ID routing is used for configuration request TLPs, completion TLPs, and optionally vendor-defined messages. The TLP header shown is only 12 bytes, but some ID routed packets can have a 16 byte header depending on the TLP type.



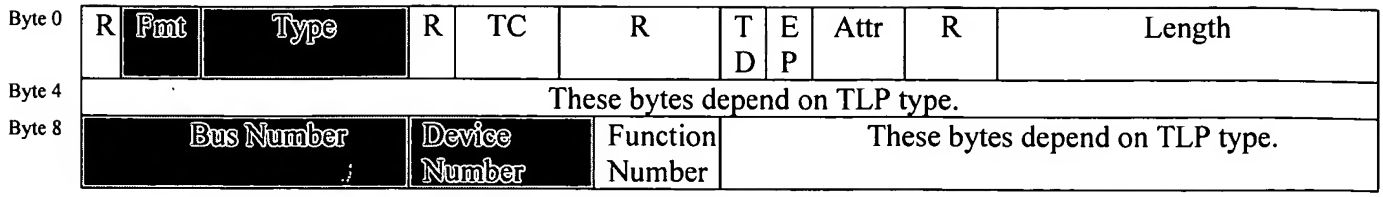


Figure 2.2-7: Header format for ID-based Transactions

For this type of transaction, the bus_number[7:0] field is used to determine which port a TLP needs to be sent to. If the bus_number field indicates one of our switch's VP2P bridge headers is being addressed, the COP will use the device number field to determine which header is being addressed and take the appropriate action.

The header fields are shown in the table below for these transaction types:

TLP Type	Fmt[1:0]	Type[4:0]	Description
CfgRd0	00	0 0100	Configuration Read Type 0
CfgWr0	10	0 0100	Configuration Write Type 0
CfgRd1	00	0 0101	Configuration Read Type 1
CfgWr1	10	0 0101	Configuration Write Type 1
Msg	01	1 0010	Message Request – Routed by ID
MsgD	11	1 0010	Message Request with data payload – routed by ID
Cpl	00	0 1010	Completion Without Data –Used for I/O and Configuration Write Completions and Read Completions (I/O, Configuration, or Memory) with Completion Status other than Successful Completion
CplD	10	0 1010	Completion with Data – Used for Memory, I/O, and Configuration Read Completions.
CplLk	00	0 1011	Completion for Locked Memory Read without Data – Used only in error case.
CplDLk	10	0 1011	Completion for Locked Memory Read – otherwise like CplD.

Table 2.2-3:

Note that configuration requests will always be sent to the COP for processing. The function_number fields shouldn't need to be used by the switch since we won't support any multi-function devices.

For messages and completions, the bus number is compared against the programmed secondary and subordinate bus numbers for the port's P2P bridge header. An example is shown below from a software topology point of view.

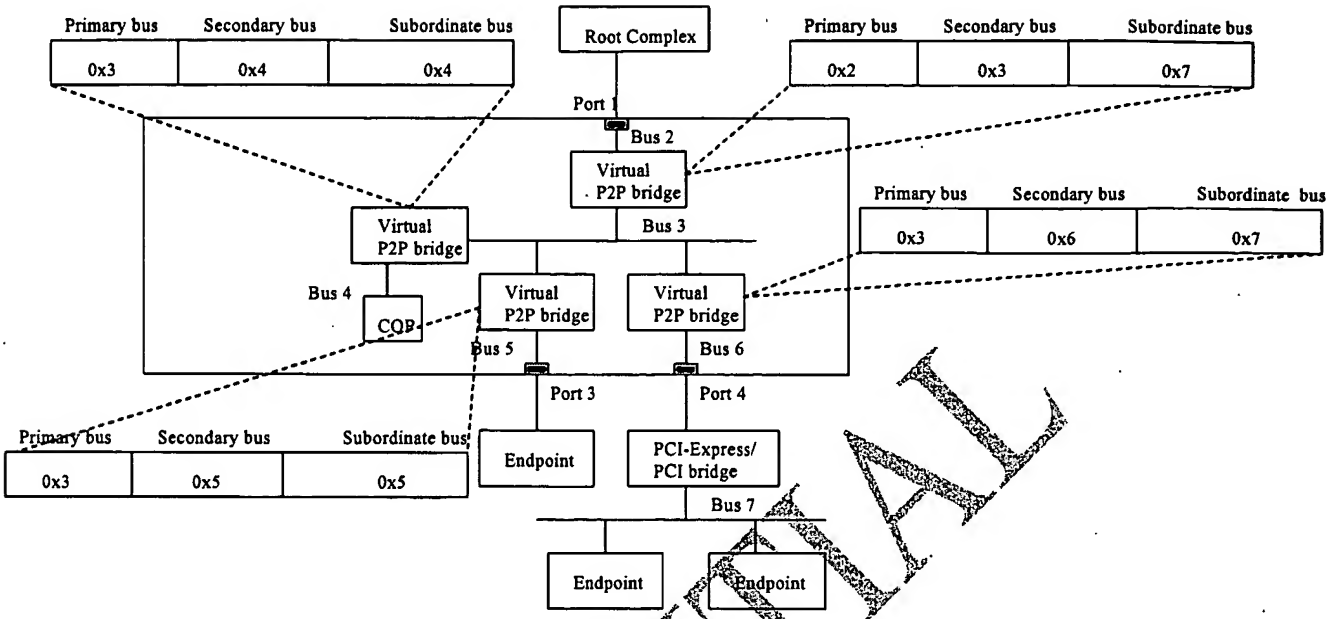


Figure 2.2-8:

Example 1

A configuration packet arrives on port 1.

- If the packet is a type 0 configuration packet, the Address_lookup_module immediately returns port 16 as the destination since this packet will be handled by the COP.
- If the packet is a type 1 configuration packet, the Bus Number field of the header is compared with Port 1's P2P bridge secondary and subordinate bus numbers. If (bus number <3) or (bus number > 7), the transaction is dropped and a UR transaction is signaled back to the TX MAC since it isn't in the range of the {secondary,subordinate} tuple.
- Else if the Bus Number field is equal to 3 (the secondary bus number of the P2P bridge header), the packet is addressing one of the downstream virtual P2P bridges inside the switch. Again, the Address_lookup_module returns port 16, and the packet is sent to the COP.
- Else, the {secondary,subordinate} tuples of all 16 destination ports are compared to see which port this packet should be routed to (ie, secondary_bus[port 3] < packet_bus_number <= subordinate_bus[port 3], etc.). Once a match is found, the destination port is returned. Note this could still return port 16 which would mean the BIOS is addressing the COP's type 0 header space.

2.2.6 Implicit routing

Message transaction are the only type that can use implicit routing, and the port logic should always examine the r[2:0] sub-field for message packets to see how to handle them. The following table shows the decoding of the various values.

r[2:0]	Definition

000	Routed to Root Complex
001	Routed by Address
010	Routed by ID
011	Broadcast from Root Complex
100	Local – Terminate at Receiver
101	Gathered and Routed to Root Complex (see section 5.3.3.2.1 of Base Spec)
110-111	Reserved – Terminate at Receiver

Table 2.2-4:

If a messages transaction's value of r[2:0] is either 001 (address routing) or 010 (ID routing), the port logic will follow the same sequence of events listed above in sections 2.2.4 and 2.2.5. If the value is one of the other values, the route is "implicit" since the definition of the encoding tells the port logic what to do.

2.2.6.1.1 Routed to Root complex

Our chip will have a root_complex field assigned to each input port based on the source OSD to handle this type of packet. The Address_lookup_module will simply index the root table (by OSD number) and return the root port number to the MAC.

2.2.6.1.2 Broadcast from Root Complex

This only applies to two types of TLPs, PME_Turn_Off and Unlock. Unlock will not actually be broadcast (which is legal according to the spec) since our switch will track which port has been locked and unlock the appropriate queues.

PME_Turn_Off will be handled by the address_lookup_module logic and the COP. The lookup logic

2.2.6.1.3 Local – Terminate at Receiver

This type of packet will always be returned from the Address_lookup_module as routed to port 16 for the COP to handle (ports 0 through 15 are the actual data ports).

2.2.6.1.4 Gathered and Routed to Root Complex

This type of routing is only used whenever the switch receives a PME_TO_Ack messages from downstream ports. The Address_lookup_module will again return port 16. The COP will scoreboard the responses from all downstream ports. Once the COP has received a PME_TO_Ack packet from each downstream port, it then returns a single PME_TO_Ack packet back to the root complex and sets the r[2:0] sub-field to 3'b101 to tell the root complex that all downstream ports in the switch responded to the PME_Turn_Off message.

Note that a timer must also be implemented in this logic to avoid deadlock, since the return of the PMC_TO_Ack packet back to the Root should not be blocked due to one device's failure to send a PME_TO_Ack in a reasonable amount of time (no time given in the spec for this, but 100 ms is mentioned elsewhere as a timeout number, so maybe we'll use that).

2.2.6.1.5 Reserved – Terminate at Receiver

These are nearly identical to the Local – Terminate at Receiver types of routing. This transaction will return port 16 from the Address_lookup_module and will be sent to the COP.

2.2.7 Isolation of OS Domains

As TLPs are received on a Rx MAC, they are queued into a structure that is separated by OS domain. PCI-EX ordering rules are always adhered to within an OS domain but never across OS domains to avoid head-of-line blocking conditions. Flow control is also isolated by OS domain and VC. The PCI-EX base specification defines flow control on a per VC basis. The switch is configured to allow assignment of VC resources per OS domain to enable absolute flow control (i.e. flow control per VC per OS domain). Therefore, from the Rx MAC perspective, each OS domain is allotted its own buffer, queue, and flow control resources.

In the Switch Core, all transactions are arbitrated using a simple round-robin algorithm, where fairness is enforced at 3 different levels – port arbitration, VC arbitration, and OSD arbitration.

The COP manages the different OS domains by appearing as a Type 0 "shared" I/O device. After switch configuration is completed, each Root Port is assigned to a particular OS domain, and for each Root Port bus, what ports are targets on that bus and their corresponding port/OSD. Any one Root Port has no knowledge of the presence of any other Root Port. If a Root Port bus targets a shared I/O port, it has no knowledge of the other OS domains that can access the shared I/O port. Therefore, at the COP level, OS domains are completely isolated.

One exception to the above rule is a scenario where the console management software is running on a trusted blade server. In that case, the OSD is allowed to manage the device-specific registers by accessing them through the COP's type 0 header space. To become a trusted blade server, the driver on that OSD must present a key to the switch. If that key matches the trusted key, the COP sets a bit that tells that OSD it can now access our device-specific registers.

2.3 Data Flow Examples

2.3.1 Address-based Requests

All Memory transactions (Reads, Writes, and Completions) and I/O transactions (Reads, Writes, and Completions) are address-based. The following describes the data flow of an address-based TLP:

1. The initiator generates a Posted Request.
2. The Rx MAC receives the Posted Request from the SERDES.
3. The Rx Physical Layer Module (RxPHY) performs 8b/10b decoding, de-scrambles the data stream, and performs clock compensation between the

extracted clock and the core clock. The data stream is in the form of 8 bits per clock (at 250 MHz.). It will also perform lane-to-lane deskew if lane width is greater than one.

4. The Rx Data Link Layer Module (RxDLLM) converts the data from the RxPHY to a 64-bit data path at 250MHz regardless of the link width. For example, if the link width is x1, it will take 8 clks to gather 64 bits at 250 MHz. The Rx DLLM then checks the Sequence Number and LCRC as the TLP is passing through to the Transaction Layer. The Rx DLLM also removes the Sequence Number and the LCRC from the TLP before it forwards the packet to the Rx Transaction Layer Module (TLM).
5. The RxTLM performs TC to VC mapping and OS domain identification to determine if there are enough header and data resources for the particular TLP. If there is, the TLP is stored in the header buffer and the data buffer (if the TLP contains payload) and the starting address of the Header is forwarded to the Address Lookup FIFO. All flow control calculations are implemented in the Rx TLM and are scheduled to transmit every time a TLP is stored or a FC credit is de-allocated.
6. The Rx TLM stores the header address of each TLP in the Address Lookup FIFO. When the Address Lookup Interface is ready to present a transaction to the Address Lookup Module in the Switch Core, it uses the header address stored in the Address Lookup FIFO to access the routing information (in this case, the 32-bit or 64-bit address) from the Header Buffer. The following information is passed into the Lookup Module:
    - address[43:0] – contains either the upper 44 bits of a 64-bit memory transaction, the upper 12 bits of a 32-bit memory transaction, or the upper 20 bits of an I/O transaction.
    - lookup_type[2:0] – field is used to specify the transaction type as shown in the following table. The types relative to this transaction type are highlighted:

Lookup_type[2:0]	Transaction definition
3'b000	32-bit memory transaction
3'b001	64-bit memory transaction
3'b010	ID-based transaction
3'b011	32-bit I/O transaction
3'b100	Routed to root complex
3'b101	Broadcast from root complex
3'b110	Terminate at receiver
3'b111	Reserved

    - port_is_downstream – lets the Address Lookup Module know what the most likely lookup sequence should be (routed to root complex).
    - tc[2:0] – used to help determine the egress_qid[3:0] for this transaction.
    - osd[3:0] – this is used in conjunction with the tc[2:0] field to determine the egress_qid[3:0].

Fast Path
If the Address Lookup FIFO is empty, the routing information is immediately presented to the Address Lookup Module in the Switch Core.

7. The Address Lookup Module(ALM) figures out the root port and which ports are connected to this ingress port. Then it begins walking the entries (up to 16) to find the base/limit pair that matches the address range, based on the array of valid ports to search. Note that the root entry is searched first if the 'port_is_downstream' bit is set. The ALM also determines the egress QID number (ranging from 0 to 15).
8. The ALM returns the egress_qid[3:0] and destination_port[4:0] to the Address Lookup Interface in the RxTLM. The transaction will not be submitted to the Transaction Queues until there is enough data, or until the entire packet is stored when the egress_cut_through_enbl is not set.
9. When the Address Lookup Module returns the egress_qid[3:0] and destination_port[4:0], the Address Lookup Interface stores the information in a 32 deep FIFO. This ALM Response FIFO is necessary in case the Transaction Queues are not able to accept the transactions as fast as the Address Lookup Module Interface is able to submit them. When this FIFO becomes half full, it backpressures the Address Lookup Module Interface. This means that the Address Lookup Module Interface will not issue any more requests to the Address Lookup Module until the ALM Response FIFO is less than half full.
10. The state of each transaction queue (empty or non-empty) in the Rx PM is sent to the Switch Core. The Switch Core will query the Presentation Module asking for the next transaction in a particular queue. The Presentation Module will resolve the transaction ordering and return a packet header to the Switch Core. The Switch Core will check flow control credits and will queue the transaction unless it does not pass flow control gating. The Switch Core will use the query information to eventually request the Presentation Module to transmit a packet.
11. The Presentation Layer will read the packet information and pop the packet out of the Transaction Queue. It will pass the packet information to the packet scheduler.
12. The Packet Scheduler will take the packet information, and create a TLP by taking the original TLP header and appending the TLP data. This packet is sent to the Data Mover.
13. Once the Switch Data Mover starts transferring the data, the TLP is stored in the Tx TLM Mini-FIFO 64 bits at a time. As the TLP is read from the Mini-FIFO, the Shared IO Header is inserted before the TLP Base Header (if the endpoint is a shared I/O). This is fed into the Tx Data Link Layer Module (DLLM).
14. The Tx DLLM receives the TLP from the Mini-FIFO and starts calculating the LCRC along with appending the Sequence Number to the start of the TLP.
15. The TLP is forwarded to the Tx Physical Layer Module (PLM) where it is scrambled and decoded from 8bits to 10bits and sent out on the wire.

2.3.2 Configuration Requests and Completions

Configuration Request and Completions follow the same steps in the previous section except that they are ID based transactions. Steps 6 and 7 would be replaced with:

6. The Rx TLM stores the header address of each TLP in the Address Lookup FIFO. When the Address Lookup Interface is ready to present a transaction to the Address Lookup Module in the Switch Core, it uses the header address stored in the Address Lookup FIFO to access the routing information (in this case, the bus number) from the Header Buffer. The following information is passed into the Lookup Module:
   - address[43:0] – contains the 8-bit bus number.
   - lookup_type[2:0] – field is used to specify the transaction type as shown in the following table. The types relative to this transaction type are highlighted:

Lookup_type[2:0]	Transaction definition
3'b000	32-bit memory transaction
3'b001	64-bit memory transaction
3'b010	ID-based transaction
3'b011	32-bit I/O transaction
3'b100	Routed to root complex
3'b101	Broadcast from root complex
3'b110	Terminate at receiver
3'b111	Reserved

   - port_is_downstream – lets the Address Lookup Module know what the most likely lookup sequence should be (routed to root complex).
   - tc[2:0] – used to help determine the egress_qid[3:0] for this transaction.
   - osd[3:0] – this is used in conjunction with the tc[2:0] field to determine the egress_qid[3:0].

> Fast Path
> If the Address Lookup FIFO is empty, the routing information is immediately presented to the Address Lookup Module in the Switch Core.

7. The Address Lookup Module(ALM) figures out the root port and which ports are connected to this ingress port. Then it begins walking the entries (up to 16) in t he bus_number_lookup_table to find the base/limit pair that matches the address range, based on the array of valid ports to search. Note that the root entry is searched first if the 'port_is_downstream' bit is set. The ALM also determines the egress QID number (ranging from 0 to 15). If the ALM determines that a particular configuration request is targeted for the COP, it will notify the Address Lookup Interface that the TLP configuration type should be changed from type 1 to type 0 when the TLP is forwarded to the Data Mover.

2.3.3 Message Requests

Each type of Message Request behaves differently regarding data flow. The following sections describe the various types of Message Requests and how it differs from the data flow example in Section 2.3.1.

2.3.3.1.1 INTx Interrupt Signaling Message Requests

The INTx virtual wire interrupt signaling mechanism is used to support legacy Endpoints in cases where the Message Signaled Interrupt mechanism cannot be used. All INTx messages are routed to the Root Complex.

INTx Messages follow the same steps in the previous section except for steps 6 and 7 would be replaced with:

6. The Rx TLM stores the header address of each TLP in the Address Lookup FIFO. When the Address Lookup Interface is ready to present a transaction to the Address Lookup Module in the Switch Core, it uses the header address stored in the Address Lookup FIFO to access the routing information (in this case, the bus number) from the Header Buffer. The following information is passed into the Lookup Module:
   - address[43:0] – contains the 8-bit bus number.
   - lookup_type[2:0] – field is used to specify the transaction type as shown in the following table. The types relative to this transaction type are highlighted:

Lookup_type[2:0]	Transaction definition
3'b000	32-bit memory transaction
3'b001	64-bit memory transaction
3'b010	ID-based transaction
3'b011	32-bit I/O transaction
3'b100	Routed to root complex
3'b101	Broadcast from root complex
3'b110	Terminate at receiver
3'b111	Reserved

   - port_is_downstream – lets the Address Lookup Module know what the most likely lookup sequence should be (routed to root complex).
   - tc[2:0] – used to help determine the egress_qid[3:0] for this transaction.
   - osd[3:0] – this is used in conjunction with the tc[2:0] field to determine the egress_qid[3:0].

> Fast Path
> If the Address Lookup FIFO is empty, the routing information is immediately presented to the Address Lookup Module in the Switch Core.

7. In this case, the Address Lookup Module(ALM) knows that the TLP is going to a Root Complex, so it only has to figure out the root port. Then it uses the device number to determine the mapping of the INTx virtual wire on primary side of bridge. The ALM returns int_map[1:0] to the Address Lookup Interface which stores it in the Header Info Code in the Header Buffer so that the Packet Generator will know to overwrite the Code in the TLP with the Code provided in the Header Info Code.

2.3.3.1.2 Power Management Message Requests

There are two Power Management Messages that require special handling in the Address Lookup Interface in the Rx TLM. They are PME_Turn_Off and PME_TO_Ack.

2.3.3.1.3 PME_Turn_Off

PME_Turn_Off is generated by a root complex to notify all of its downstream ports to prepare for power removal. PME_Turn_Off Message has a routing type of 3'b101 which is 'broadcast from root complex'. In this case, steps 6 and 7 are replaced by the following:

6. The Rx TLM stores the header address of each TLP in the Address Lookup FIFO. When the Address Lookup Interface is ready to present a transaction to the Address Lookup Module in the Switch Core, it uses the header address stored in the Address Lookup FIFO to access the routing information (in this case, the bus number) from the Header Buffer. The following information is passed into the Lookup Module:
   - address[43:0] – contains the 8-bit bus number.
   - lookup_type[2:0] – field is used to specify the transaction type as shown in the following table. The types relative to this transaction type are highlighted:

Lookup_type[2:0]	Transaction definition
3'b000	32-bit memory transaction
3'b001	64-bit memory transaction
3'b010	ID-based transaction
3'b011	32-bit I/O transaction
3'b100	Routed to root complex
3'b101	**Broadcast from root complex**
3'b110	Terminate at receiver
3'b111	Reserved

   - port_is_downstream – lets the Address Lookup Module know what the most likely lookup sequence should be (routed to root complex).
   - tc[2:0] – used to help determine the egress_qid[3:0] for this transaction.
   - osd[3:0] – this is used in conjunction with the tc[2:0] field to determine the egress_qid[3:0].

> Fast Path
> If the Address Lookup FIFO is empty, the routing information is immediately presented to the Address Lookup Module in the Switch Core.

7. In this case, the Address Lookup Module(ALM) knows that the TLP needs to be broadcast to all downstream ports configured to the Root Complex, so it looks up the endpoints that the TLP should be sent to by asserting the corresponding bits in broadcast_ports[15:0]. The Address Lookup Interface then submits a TLP to the Transaction Scheduler for each downstream port designated in

broadcast_ports[15:0], along with sending it to the cop (port16). The Address Lookup Interface will then halt all forward progress until it has been notified by the COP that the scoreboard is set up and it is ready to receive all the PME_Turn_Off Messages from each downstream port.

2.3.3.1.4 PME_TO_Ack

The only thing to note for PME_TO_Ack Messages is that when an upstream port sends a PME_TO_Ack message, they should all be routed to the COP. The COP will keep a scoreboard of all the endpoint ports that received a PME_Turn_Off Message as they transmit a PME_TO_Ack message. When all of the endpoint ports have transmitted a PME_TO_Ack message (or the timer expires), the COP will generate one PME_TO_Ack Message to the Root Complex.

2.3.3.1.5 Locked Transaction Message Requests

Whenever a particular root requests a locked transaction, all other sources going to that output will be halted. When the CplLk is received from the downstream port, all other upstream queues going to the root are locked until the Unlock message is received.

2.4 Shared Link Descriptions

2.4.1 AS Encapsulation

AS header encapsulation only pertains to Shared Ports. Non-shared Ports have no knowledge of "Shared I/O". On the Rx MAC, the Transaction Layer Module strips the AS Header from all TLPs before it stores the packet in the in-line buffer. It uses the AS Header to determine the OS Domain associated with the given TLP. On the Tx MAC, the Transaction Layer Module inserts the AS Header to the given TLP. The OS Domain that is inserted as part of the AS Header is reported by the Switch Core and passed on to the Tx Transaction Layer Module.

The AS Header is described in detail in the follow section. Figure 2.4-2 describes the format of the AS Header.

2.4.2 OS Domain Routing

For our switch, ports can be shareable ports, which means multiple different CPUs can address resources over the same PCI-Express link. A maximum of 16 OS Domains (or CPUs) will be supported in this implementation, with each port having the capability to send and receive from 16 OS Domains, across all 8 VCs possible in the PCI Express spec.

The PCI-Express Advanced Switching (AS) spec incorporates a 8-byte AS header that is inserted into the transaction. Our switch will use this header to specify the OS Domain

associated with a given transaction. The following diagram shows where this header is located in the packet.

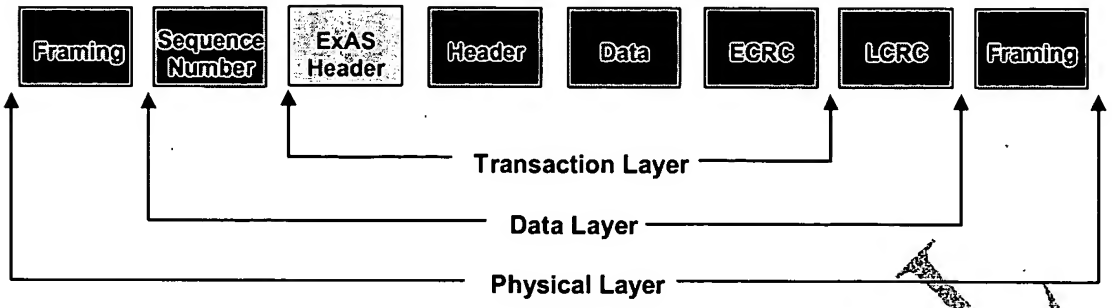


Figure 2.4-1:

AS headers specify an 8-bit field, called the PI (Protocol Interface) field, that defines what type of AS packet is contained in the payload. We're free to use any number ranging from 224 through 254 as a vendor-defined PI. The only other piece of information in our AS header will be the 8-bit OS Domain tag. The proposed AS header is shown below.

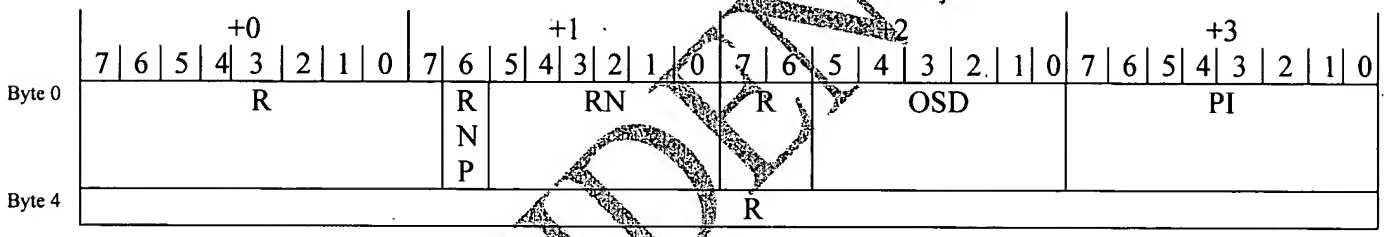


Figure 2.4-2:

PI – Protocol Identifier field from AS
OSD – OS Domain number
RNP – Resource Number Present (when high, the RN field is valid.. when low, it is invalid and must be set to all 0's)
RN – Resource Number (which buffer this packet belongs to)
R – reserved

On shared ports, the RX MAC will first check the PI field to make sure this type of AS packet is understood by our switch. We'll have a register within the RX MAC defined that contains the allowable encoding for this type of AS packet. At first, this will likely be our selected vendor-defined PI number. If our technology is adopted by the SIG, a standard number will be defined, and this register will then be programmed to contain this standard value for use with future devices.

This AS header allows our switch to map the incoming value of the OS Domain field (local to the I/O device or inter-switch port) in the AS header to the global OS Domain number within the switch (one of 16 values, 0 through 15 for the first revision of our switch). Any packets received on the shared port that use a different PI type will be discarded.

For upstream ports, the RX MAC will only need to associate the OSD number with a 4-bit register number loaded at config time. This number will be the OSD field that is passed to the lookup_unit to match the correct set of P2P bridge headers to compare for a lookup match.

2.4.3 Flow Control and Credits

Flow Control is used to track the queue/buffer space available in the Agent across the link. It is used to prevent overflow of receiver buffers. The Flow Control information is conveyed between two sides of the Link using DLLP packets.

For Non-shared Endpoints (only one OSD), flow control updates follow the same format as the PCI Express Base Specification, which is as follows:

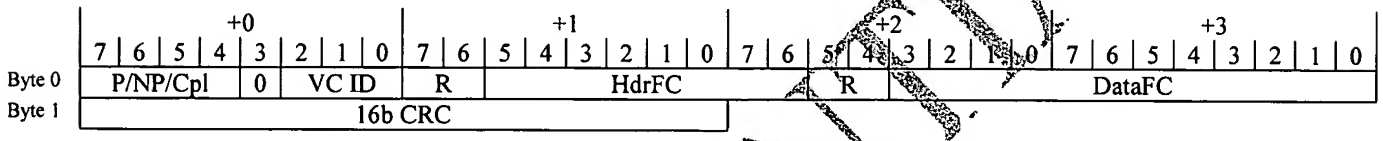


Figure 2.4-3: DLLP format for Flow Control Packet for Non-Shared Ports

P/NP/Cpl: This field specifies the type of transaction that is being reported. P – Posted Request; NP – Non-posted Request; Cpl – Completion.
VC ID: This field specifies the Virtual Channel that is being reported.
R: Reserved
HdrFC: This field contains the credit value for Headers of the indicated type. One credit value for headers is one maximum-size header plus TLP digest.
DataFC: This field contains the credit value for payload data of the indicated type. One credit value is equivalent to 16 bytes of data.
16bCRC: This field contains the calculated CRC value of all bits of the packet using the polynomial coefficient of 100Bh.

For Shared Endpoints, flow control updates are advertised using the following DLLP to account for the OSD:

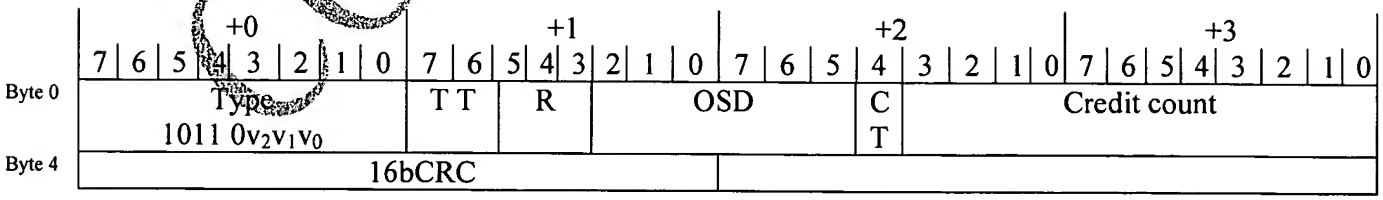


Figure 2.4-4: DLLP format for Flow Control Packet for Shared Ports

- Type: Upper nibble set to 1011 for an FC Update shared-link DLLP. The lower nibble specifies the VC number.
- TT: Transaction Type (00 for Posted, 01 for Non-posted, 10 for Completions)
- R: Reserved

- OSD: OSD number
- CT: Credit Type (0 for header credits, 1 for data credits
- Credit count: Contains either the 12-bit data credit count or the 8-bit (upper 4 bits are zeros) header credit count, based on the value of the CT bit
- 16bCRC: This field contains the calculated CRC value of all bits of the packet using the polynomial coefficient of 100Bh.

2.4.3.1.1 Receiver Flow Control

The RxTLM keeps track of flow control accounting functions of its buffers. This information is assembled into a FCUpdate DLLP and forwarded to the Tx Transaction Layer Module where it is scheduled to be transmitted to the Agent across the link.

For each type of information tracked, the following quantities are calculated for flow control TLP Receiver accounting (for non-shared ports, these calculations are performed for each VC, and for shared ports these calculations are performed for each VC/OSD group):

- CREDITS_ALLOCATED – The total number of credits granted to the Transmitter since initialization, modulo $2^{[\text{Field Size}]}$ (where [Field Size] is 8 for headers and 12 for payload data).
- CREDITS_RECEIVED – The total number of FC units consumed by valid TLPs received since flow control initialization, modulo $2^{[\text{Field Size}]}$ (where [Field Size] is 8 for headers and 12 for payload data).

The RxTLM will also check for buffer overruns. This is done by checking the following equation:

$$(\text{CREDITS\_ALLOCATED} - \text{CREDITS\_RECEIVED}) \text{ modulo } 2^{[\text{Field Size}]} \geq 2^{[\text{Field Size}]}/2$$

The scheduling of transmission of UpdateFC DLLPs will obey the following rules:

- If the Link is in the L0 or L0s Link state, UpdateFC DLLPs will be scheduled for transmission once every 30us or 120us, depending on the status of the Extended Sync bit in the Control Link Register.
    - A Timer will also be implemented with the following rules:
        - The Timer is active only when the Link is in the L0 or L0s Link state.
        - The Timer has a limit of 200us.
        - The receipt of any Init or Update FC DLLP resets the Timer.
        - Upon Timer expiration, the Physical Layer will be instructed to retrain the Link.

Otherwise, for all types of transactions that do not have infinite credit, a Flow Control DLLP will be scheduled for transmission after a valid TLP is received and stored, or when one unit is made available by TLPs processed.

2.4.3.1.2 Transmitter Flow Control

The Transaction Scheduler receives the latest FC updates from all the Tx MACs. These FC updates report the most recent number of FC units advertised by the receiver on the other side of the link called CREDIT_LIMIT. CREDIT_LIMIT is used to determine if the transactions being transferred to a particular Tx MAC has enough FC credits to be forwarded to the appropriate Tx MAC.

For each type of information tracked, the following quantities are calculated for flow control gating:

- CREDITS_CONSUMED – The total number of FC units consumed by TLP transmissions made since flow control initialization, modulo $2^{[\text{Field Size}]}$ (where [Field Size] is 8 for headers and 12 for payload data).
- CREDIT_LIMIT – The most recent number of FC units legally advertised by the Receiver. This quantity represents the total number of FC credits made available by the Receiver since flow control initialization, modulo $2^{[\text{Field Size}]}$ (where [Field Size] is 8 for headers and 12 for payload data).

To determine if there is enough credit for the current transaction, the following equation is evaluated:

(CREDIT_LIMIT – (CREDITS_CONSUMED + *current_tlp_credits_needed*)) modulo $2^{[\text{Field Size}]} \leq 2^{[\text{Field Size}]}/2$

Even though the Transaction Scheduler determines Flow Control gating of transactions, it does not have any knowledge of the error status of the transaction. It is not until the Transaction Scheduler forwards the TLP to the Tx MAC that it is known if the packet is in error. Therefore, the Tx MAC is responsible for notifying the Transaction Scheduler that the current TLP is in error so that it does not affect CREDITS_CONSUMED.

2.4.4 Reset

There are two types of Reset on the chip – Fundamental Reset and Hot Reset. The following diagram will be used throughout the document to describe the devices affected when a type of Reset is asserted at a Root Complex, a Root Port, the Switch, a Downstream Port, or an Endpoint (I/O device) attached to a Downstream Port. Ports 1, 2, 3, 9 and 10 are all attached to root complexes and therefore represent one OS domain (for simplicity, the OS domain number will directly correlate to the port number, i.e. port 1 is assigned OS domain 1 in this example). Switch #1 has been configured such that port 4 is shared by OS domains 1 and 3, port 5 is only accessed by OS Domain 2, port 6 is shared by OS domains 1, 2, and 3, and port 7 is shared by OS domains 1 and 2. Downstream port 4 in Switch #1 is connected to Root Port 8 in Switch #2, which enables access to the endpoints on Switch #2 from the Root Complexes in Switch #1.

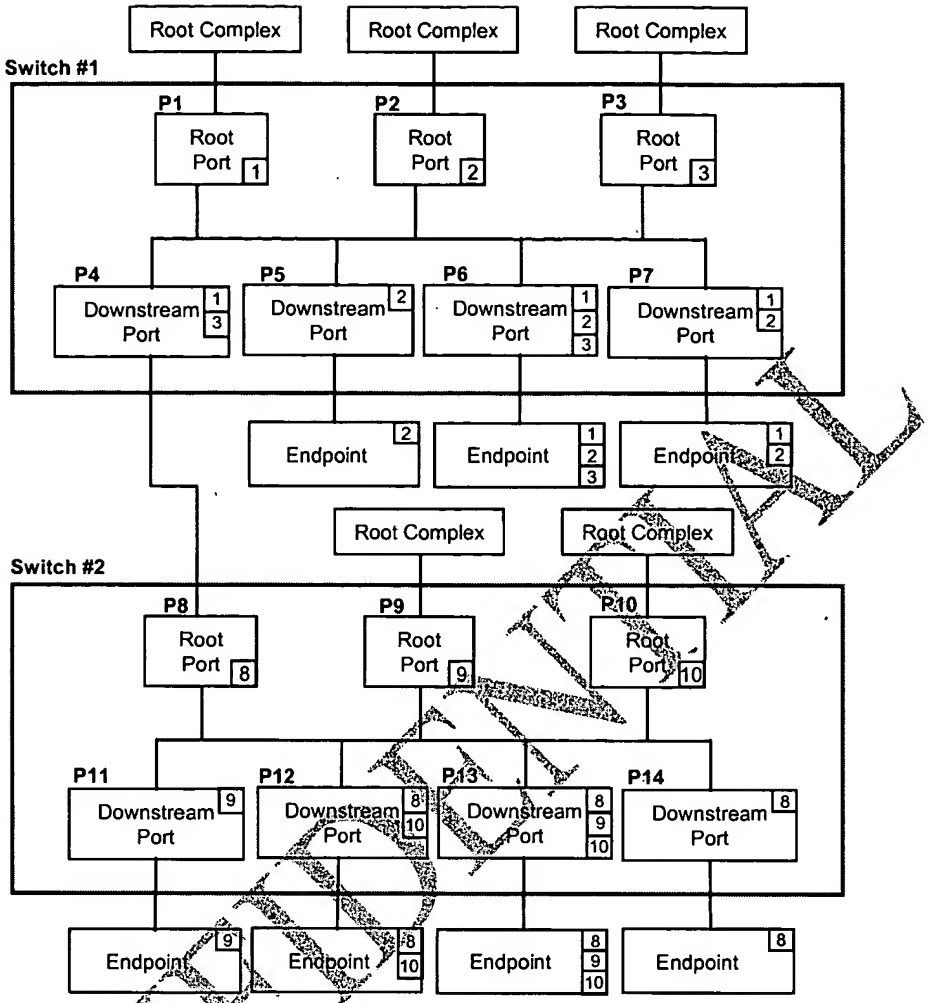


Figure 2.4-5: Example Topology

2.4.4.1.1 Fundamental Reset

Fundamental Reset is an auxiliary signal provided by the system to a component or add-in card. The signal must be called PERST#. Fundamental Reset can be asserted on the Root Complexes, the Endpoints (I/O Devices), or the Switch.

When Fundamental Reset is asserted on the Endpoints or the Switch, the behavior is identical to what is described in the PCI Express Base Specification – all of the Links attached to the device being reset will be retrained, the state machines will be initialized, and all TLP information will be flushed.

If Fundamental Reset is asserted on a Root Complex, not only does the Root Complex get reset, but all of its downstream ports must reset as well. If its downstream ports are "shared" with other Root Complexes, it is important to be able to reset only the part of the downstream port that pertains to the Root Complex being reset, and to leave the rest of the downstream port logic unchanged. This is done by transmitting and generating

vendor-specific DLLPs called Reset DLLPs that informs the two components on the link which OS Domain is getting reset (this is explained further in the following sections). If the downstream port is not shared by Root Complexes, the Link will reset according to the PCI Express Base Specification. The following sections describe how the chip behaves when a Fundamental Reset is asserted on the various parts of the chip.

2.4.4.1.2 Fundamental Reset initiated at the Switch

If the Fundamental Reset on the Switch is asserted, it will propagate the Reset to all upstream and downstream ports. The devices attached to all the ports will be reset according to the PCI Express Base Specification - all of the Links attached to the device being reset will be retrained, the state machines will be initialized, and all TLP information will be flushed. If the fabric topology involves more than one Switch, and a Root Port in another Switch is affected by the reset, then all of the downstream ports assigned to that Root Port are also reset. The components highlighted in yellow in Figure 2.4-6 depict the components that are affected when Switch #1 is reset. Everything inside Switch #1 is reset, along with the devices attached to the ports. Note that on Switch #2, only one of the root ports is affected by the reset. That scenario is explained in detail in Section 2.4.4.1.3.

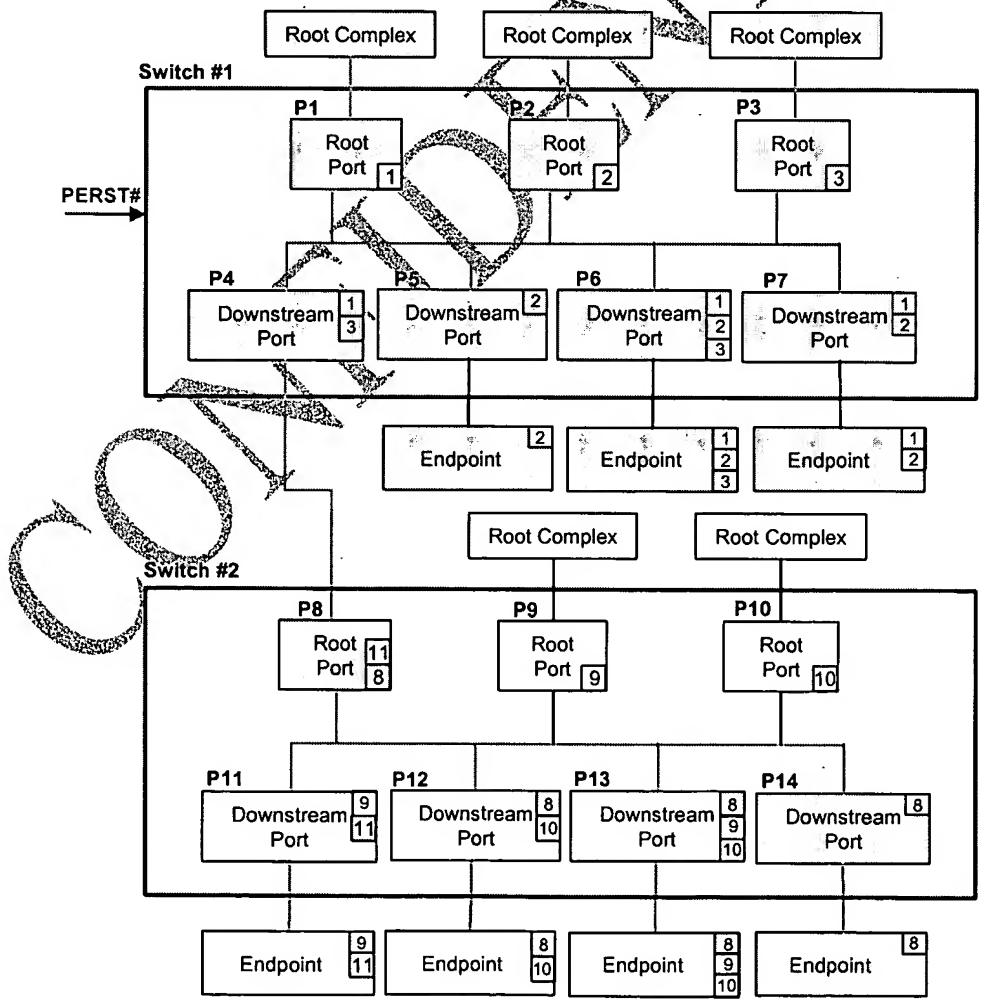


Figure 2.4-6: Fundamental Reset initiated at Switch #1

2.4.4.1.3 Fundamental Reset initiated at the Root Complex

If the Fundamental Reset on a Root Complex is asserted, the reset must be propagated to all its downstream ports without corrupting the traffic from the other OS Domains (or Root Complexes). The following steps are taken when a Fundamental Reset is asserted at a Root Complex:

At the Root Complex

1. All Port Registers and State Machines must be set to their initial values as specified in the PCI Express Base Document.
2. The Root Complex will attempt to retrain the link.
3. Once both components on the link have entered the initial link training state, they will proceed through Link Initialization and then through Flow Control Initialization for VC0.

At the Switch

1. The Root Port connected to the Root Complex will retrain and initialize all its state machines and registers.
2. The Root Port will notify the COP that all of its downstream ports need to be reset.
3. The COP will pass the reset notification to all the downstream Ports.
4. All registers and state machines relevant to the OSD being reset must be set to their initial values. All TLPs belonging to the OSD being reset must be flushed – all TLPs stored in the Rx in-line buffers will naturally drain and all TLPs in the Tx retry buffers will drain after Ack DLLPs are received. During reset, new TLPs belonging to the OSD will be rejected. All other TLPs will be preserved.
5. The Downstream Port will be notified of the reset condition and the OSD that has initiated the reset.

6a. If the Downstream Port is not shared (i.e. it is only accessed by one Root Complex), the port is reset by attempting to retrain the Link.

6b. If the Downstream Port is shared, all TLPs pertaining to the OSD that has initiated the reset must be flushed. Flow Control must also be updated to reflect the flushing of TLPs. A vendor-specific DLLP is generated called a Reset DLLP by the Downstream Port. The Reset DLLP contains the OSD that initiated the reset and is transmitted on the link. A Reset DLLP is transmitted every time the Transaction Arbiter selects the OSD that initiated the reset. Otherwise, the Transaction Arbiter schedules TLPs to transmit on the other OSDs that are operating normally. The Downstream Port will continue to transmit Reset DLLPs until the reset notification from the COP has been removed.

At the Endpoint

1a. If the Endpoint is not shared (i.e. it is only accessed by one Root Complex), the port is reset by attempting to retrain the Link.

1b. If the Endpoint is shared, it will receive the Reset DLLPs and clear all registers, state machines, and flush all TLPs that pertain to the OSD initiating the reset.

The device will stay in this reset mode until it stops receiving Reset DLLPs. All traffic related to the OSDs that are not being reset will operate normally.

The components highlighted in yellow in Figure 2.4-7 depict the components that are affected when the Root Complex attached to Root Port #1 is reset. Since downstream ports 4,6, and 7 are shared ports, only the logic pertaining to OS domain 1 should be affected by the reset. On the other hand, port 14 in Switch #2 is only accessed by the OS Domain being reset and can therefore reset the entire port by retraining the Link (instead of sending Reset DLLPs specific to an OS domain).

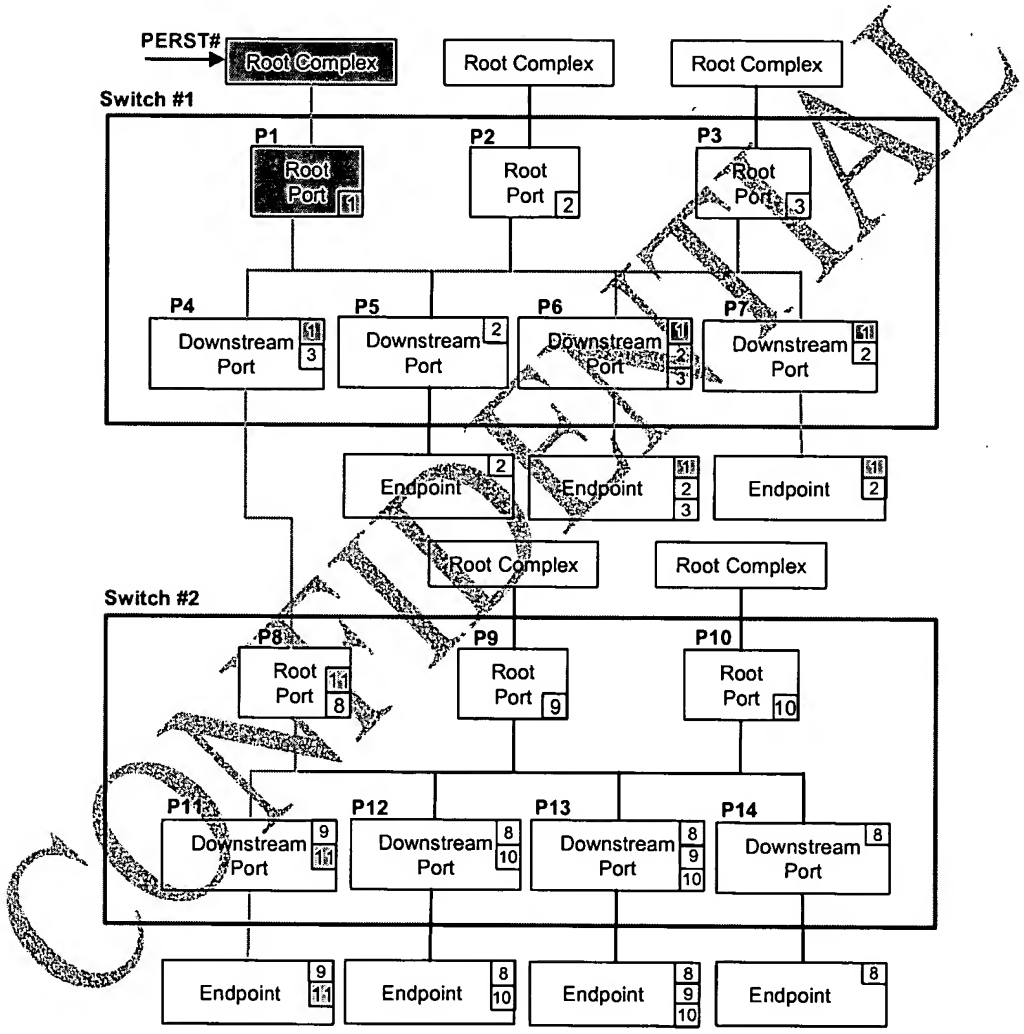


Figure 2.4-7 : Fundamental Reset initiated at a Root Complex

2.4.4.1.4 Fundamental Reset initiated at an Endpoint

If the Fundamental Reset on an endpoint device is asserted, the device will simply reset with its link according to the PCI Express Base Specification - all of the Links attached to

the device being reset will be retrained, the state machines will be initialized, and all TLP information will be flushed. All other Ports on the Switch will not be affected.

The components highlighted in yellow in Figure 2.4-7 depict the components that are affected when the endpoint connected to downstream port 13 is reset. It simply attempts to retrain with the port on the other side of the Link. Any transaction being received by the Tx MAC from the Switch Core will be discarded and will never reach the Data Link Layer Module. The Root Complex will eventually time out when it never receives a completion for a particular request. (We could also let the COP generate UR completions since it will know which endpoints are in reset. The ALM could keep track of which transactions are going to egress ports that are in reset and then route the packet to the COP.)

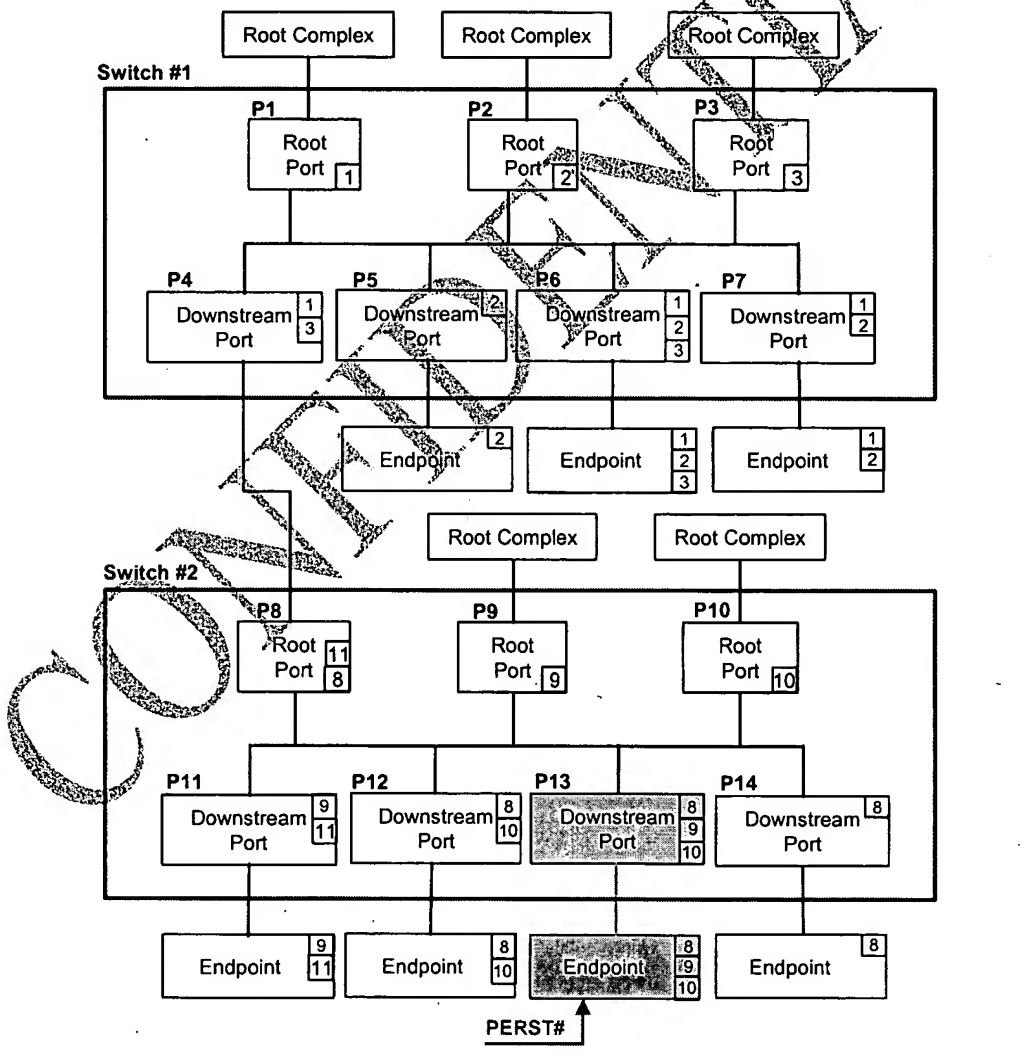


Figure 2.4-8 : Fundamental Reset initiated at an Endpoint

2.4.4.1.5 Hot Reset

Hot Reset is an in-band mechanism for propagating reset across a link. A Link can enter Hot Reset if directed by a higher layer, or if it receives two consecutive TS1 ordered sets with the Hot Reset bit asserted. The following sections describe how the chip behaves when a Hot Reset is asserted on the various parts of the chip.

2.4.4.1.6 Hot Reset initiated at the Root Port

A Root Port can enter Hot Reset by having its Secondary Bus Reset bit set in the Bridge Control Register, or by receiving two consecutive TS1s with the Hot Reset bit set on the Link. If the Hot Reset on a Root Port is initiated, the reset must be propagated to all its downstream ports without corrupting the traffic from the other OS domains (or Root Complexes). The following steps are taken when a Hot Reset is initiated at a Root Port:

At the Root Complex

1. The Root Complex receives a TS1 sequence with the Hot Reset bit asserted and will attempt to retrain the Link. This can happen if the secondary bus reset bit is set in the P2P config space.
2. All Port Registers and State Machines must be set to their initial values as specified in the PCI Express Base Document.
3. Once both components on the link have entered the initial link training state, they will proceed through Link Initialization and then through Flow Control Initialization for VC0.

At the Switch

1. The Root Port connected to the Root Complex will retrain and initialize all its state machines and registers.
2. The Upstream Port will notify the COP that all of its downstream ports need to be reset.
3. The COP will pass the reset notification to all the downstream Ports.
4. All registers and state machines relevant to the OSD being reset must be set to their initial values. All TLPs belonging to the OSD being reset must be flushed – all TLPs stored in the Rx in-line buffers will naturally drain and all TLPs in the Tx retry buffers will drain after Ack DLLPs are received. During reset, new TLPs belonging to the OSD will be rejected. All other TLPs will be preserved.
5. The Downstream Port will be notified of the reset condition and the OSD that has initiated the reset.

6a. If the Downstream Port is not shared (i.e. it is only accessed by one Root Complex), the port is reset by attempting to retrain the Link.

6b. If the Downstream Port is shared, all TLPs pertaining to the OSD that has initiated the reset must be flushed. Flow Control must also be updated to reflect the flushing of TLPs. A vendor-specific DLLP is generated called a Reset DLLP by the Downstream Port. The Reset DLLP contains the OSD that initiated the reset and is transmitted on the link. A Reset DLLP is transmitted every time the Transaction Arbiter selects the OSD that initiated the reset. Otherwise, the Transaction Arbiter schedules TLPs to transmit on the other OSDs that are

operating normally. The Downstream Port will continue to transmit Reset DLLPs until the reset notification from the COP has been removed.

At the Endpoint

1a. If the Endpoint is not shared (i.e. it is only accessed by one Root Complex), the port is reset by attempting to retrain the Link.

1b. If the Endpoint is shared, it will receive the Reset DLLPs and clear all registers, state machines, and flush all TLPs that pertain to the OSD initiating the reset. The device will stay in this reset mode until it stops receiving Reset DLLPs. All traffic related to the OSDs that are not being reset will operate normally.

2.4.4.1.7 Hot Reset initiated at the COP

A Hot Reset can be initiated at the Switch through a set of Registers that can be accessed via an I2C interface. The Hot Reset can be programmed such that it can operate on a per port and/or per OSD basis. If the Hot Reset is propagated to Ports without distinguishing between OSDs, the Ports will be reset according to the PCI Express Base Specification – all of the Links attached to the device being reset will be retrained, the state machines will be initialized, and all TLP information will be flushed. If the Hot Reset is propagated to a subset of the OS domains on a particular Port, the Port will use Reset DLLPs to reset the designated part of its port logic pertaining to the OS domain specified in the Reset DLLP.

2.4.4.1.8 Hot Reset initiated at a Downstream Port

A Downstream Port can enter Hot Reset by having its Secondary Bus Reset bit set in the Bridge Control Register. The Port will transmit Reset DLLPs on the OSDs that correspond to the Secondary Bus Reset bits that were asserted and clear all Register and State Machines pertaining to the OSD. All traffic related to the OSDs that are not being reset will operate normally.

2.4.4.1.9 Hot Reset initiated at a Shared Upstream Port

A Shared Upstream Port can enter Hot Reset by having its Secondary Bus Reset bit set in its Bridge Control Register. The Port will transmit Reset DLLPs on the OSDs that correspond to the Secondary Bus Reset bits that were asserted and clear all Register and State Machines pertaining to the OSD. All traffic related to the OSDs that are not being reset will operate normally.

2.4.4.1.10 Hot Reset initiated at an I/O Device

If a device wants to reset itself, it can do so by either transmitting Reset DLLPs or by transmitting TS1s. If the device wishes to only reset a particular OSD, it will generate and transmit Reset DLLPs that specify the OSD to reset. It will also clear all Registers and State Machines pertaining to the OSD. If the device wishes to reset the entire link, it will generate and transmit TS1s with the Hot Reset bit asserted to reset the entire link. It will also initialize all Registers and State Machines and attempt to bring up the link.

2.4.5 Power Managem nt

PCI Express Power Management provides the following services:

- It allows software driven D-state transitions to change the Link power management states for a physical Link.
- It provides a hardware-autonomous capability to change the Link power management states for a physical Link (Active State Power Management).
- It provides a wakeup mechanism driven by in-band TLPs routed from the requesting device towards the Root Complex – these are called power management event (PME) Messages.
- It provides a means to change the Power Management state by generating PMEs per PCI Express function.

2.4.5.1.1 Link State Power Management

A PCI Express physical Link can enter Link power management states by either software driven D-state transitions or by active state Link power management activities. Defined Link states include L0, L0s, L1, L2, and L3. The power savings increases as the Link state transitions from L0 through L3. Table 2.4-1 and Table 2.4-2 summarizes the Link Power Management States for both non-shared I/O and shared I/O.

	L-State Description	Used by S/W PM?	Used by ASPM?	Clocks & Power
L0	Fully Active Link	Yes (D0)	Yes (D0)	On
L0s	Standby State	No	Yes (D0)	On
L1	Lower Power Standby	Yes (D1-$D3_{hot}$)	No	On
L2/L3 Ready	Staging Point for Power Removal	Yes	No	On
L3	Off	n/a	n/a	Off

Table 2.4-1: Summary of Non-Shared I/O Link Power Management States

	L-State Description	Used by S/W PM?	Used by ASPM?	Clocks & Power	Shared I/O
L0	Fully Active Link	Yes (D0)	Yes (D0)	On	Normal Operation
L0s	Standby State	No	Yes (D0)	On	TLP & DLLP transmission is prohibited for all OSDs(ASPM only)
L1-L3	Lower Power States	No	No	On	TLP & DLLP transmission is prohibited for a specific OSD

Table 2.4-2: Summary of Shared I/O Link Power Management States

2.4.5.1.2 Power Management Software Control

One of the ways that power management states of a Link are determined is by the software driven D-state of its downstream component. Table 2.4-3 depicts the relationship between the power state of a component and its Upstream Link. A Downstream component can be an Endpoint or another Switch.

Downstream Component D-State	Permissible Upstream Component D-State	Permissible Interconnect State (for Non-Shared I/O)	Permissible Interconnect State (for Shared I/O)
D0	D0	L0, L0s	L0, L0s
D1 (optional)	D0-D1	L1	L0, L0s (Cannot go into a low level power state)
D2 (optional)	D0-D2	L1	L0, L0s (Cannot go into a low level power state)
$D3_{hot}$	D0-$D3_{hot}$	L1, L2/L3 Ready	L0, L0s (Cannot go into a low level power state)
$D3_{cold}$	D0-$D3_{cold}$	L3	L0, L0s (Cannot go into a low level power state)

Table 2.4-3: Relation between Power Management States of Link and Components

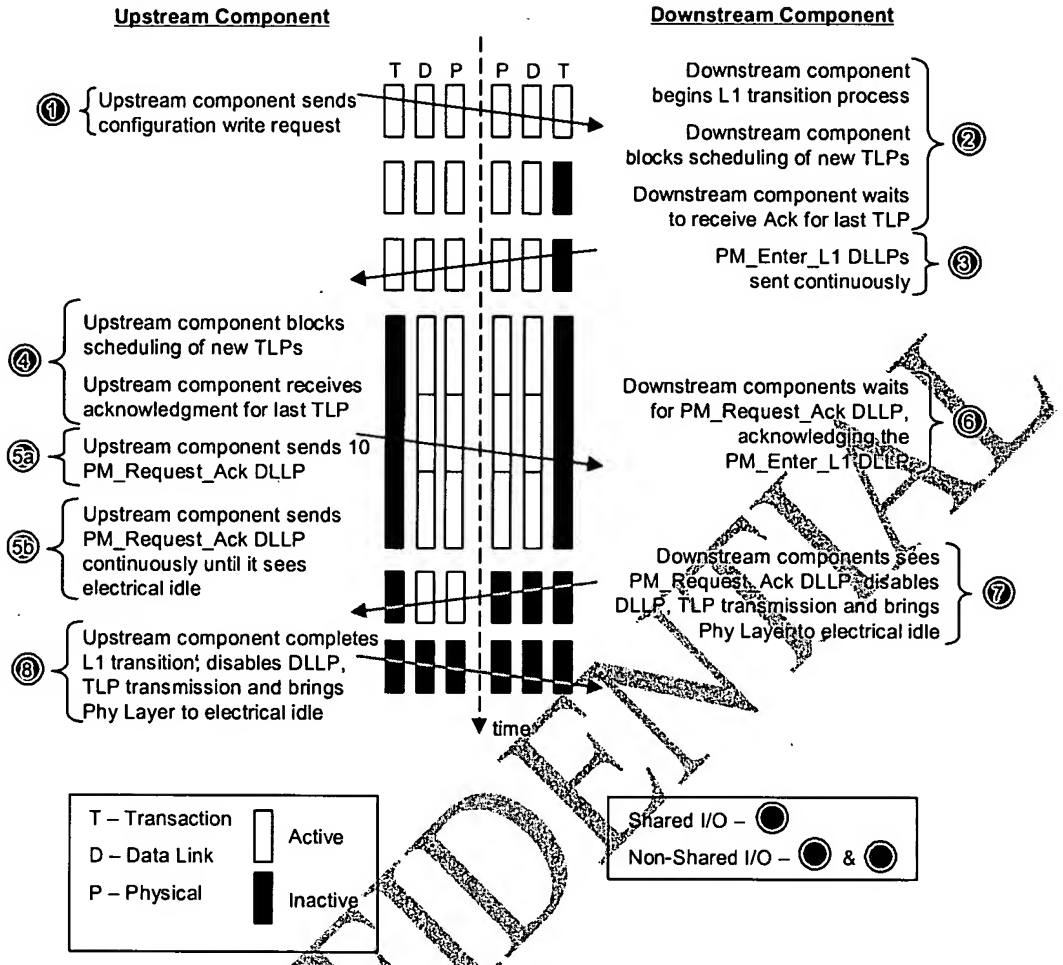


Figure 2.4-9: Entry into L1 Link State

2.4.5.1.3 Active State Power Management (ASPM)

Active State Power Management (ASPM) is an autonomous hardware based active state mechanism that enables power saving even when the connected components are in the D0 state (fully operational state). After a period of idle Link time, the ASPM mechanism engages in a Physical Layer protocol that places the idle Link into a lower power state. Once in the lower power state, transitions to the fully operative L0 state are triggered by traffic appearing on either side of the Link. This feature may be disabled by software.

Since ASPM is initiated by the link being in idle for a specified amount of time, the physical layer can be placed in a lower power state regardless of whether the component is shareable or not. When any traffic (regardless of OS domain) appears, the link is placed in the fully operative L0 State.

2.4.5.1.4 Power Management Event Mechanisms

2.4.5.1.5 Link Wakeup

PCI Express components are permitted to wakeup the system using a wakeup mechanism followed by a power management event (PME) Message. These PME Messages are in-band TLPs routed from the requesting device towards the Root Complex. This PME mechanism is broken up into two tasks:

- Reactivation (wakeup) of the associated resources
- Sending a PME Message to the Root Complex

The Link wakeup mechanisms provide a means of signaling the platform to re-establish power and reference clocks to the components within its domain. There are two defined wakeup mechanisms: Beacon and WAKE#. The Beacon uses in-band signaling to implement wakeup functionality. WAKE# is an input to the Switch, and in response to WAKE# being asserted, the Switch must generate a Beacon that is propagated to the Root Complex.

The Switch must translate the wakeup mechanism appropriately when some ports use the beacon mechanism and others use WAKE#. The COP will keep a scoreboard of the downstream ports wakeup states, and when all the downstream ports of a specific Root Complex have been woken up, the COP will either send a beacon or WAKE# to the Root Port.

Regardless of the wakeup mechanism used, once the Link has been re-activated and trained, the requesting agent then propagates a PM_PME message upstream to the Root Complex.

2.4.5.1.6 PME Messages

PCI Express devices need to be notified before their reference clock and main power is removed so that they can prepare for it. This is done as follows:

1. Before power and clocks are turned off, the Root Complex (or Downstream Port) issues a PME_Turn_Off message to all agents downstream to cease initiation of any subsequent PM_PME messages.
2. Each agent is required to respond with a PME_To_Ack TLP, which must terminate at the point of origin.
3. As each agent responds with a PME_To_Ack TLP, the TLP is received by the endpoint port and routed to the COP. When the COP receives PME_To_Ack TLPs for all of a particular Root Complex's downstream ports, the COP generates and sends a PME_To_Ack TLP to the Root Port.
4. Once an endpoint port has sent the PME_To_Ack packet, it must then prepare for removal of power and clocks by initiating a transition to the L2/L3 Ready state.
5. The Switch is responsible for making sure that the upstream port goes to L2/L3 Ready state after all its downstream ports have entered L2/L3 Ready state. It

should not wait indefinitely for the PME_To_Ack packet, but should implement a timeout mechanism where it would assume that the PME_Turn_Off TLP was received after the timeout expired.

6. The Power Delivery Manager must wait a minimum of 100ns. after the Root Complex transitioned to L2/L3 Ready before removing power and clocks.

2.4.6 Switch initialization

There are some basic events that must always happen in whenever our device powers up from a software/configuration point of view.

1. I2C initialization
2. Link training
3. OSD negotiation
4. System setup update (optional)
5. FC initialization
6. Pseudo-Device Discovery (optional)
7. Device Discovery

Each will be covered in more detail in the following sections.

2.4.6.1.1 I2C initialization

There will be hardware defaults for many of the registers in our chip that should provide a functional chip at boot time. There are, however, a few structures that must be provisioned by the I2C interface at boot time since the hardware has no idea what type of system is being created.

There will be a default binding of OSDs to port set up by the I2C. This means the I2C will set up the required mapping tables so that transactions that enter the switch know where to go for the various OSDs on that port. The following structures must be provisioned with "default" values by the I2C for the switch to boot:

- TC/VC mapping table (indexed by osd[3:0]) in the MAC: set to all 0s except for TC0 which is hardwired to 1 on VC0. Note that there is one of these tables per port.
- Destination QID RAM in the Address_lookup_module (indexed by {dest_port[4:0],osd[3:0],tc[2:0]}: All entries for dest_port=16 should be provisioned to return some 5-bit number as the dest_qid for all OSD/Tc combinations the system expects to appear. Again, the valid bit should be set for these entries. There is only one of these in the switch.

For example, if the EEPROM expects a 2-OSD device to be plugged into port 8, and this device should talk to ports 0 and 1, the I2C will write the data in the Destination QID

RAM at address {5'd16,4'd0,3'd0} to a value of 0. It will also write the address at {5'd16,4'd1,3'd0} with a value of 1. By writing these values (and setting the valid bit for each address), initial transactions will be queued in the switch core and eventually routed to the COP through the correct queue groups.

2.4.6.1.2 Link training

This protocol should follow the base specification mechanism to allow a given port to train as a 1x, 2x, 4x, or 8x link. Since our switch is actually composed of 8x and 4x PCI-Express cores (an 8x core plus a 4x core will be contained in a MAC), the 8x core in each MAC will attempt to train first. If it trains in anything less than 8x, it'll "turn on" the 4x core and allow it to attempt to train. If it trains to 8x, the 4x core will never wake up since all 8 lanes are in use by the 8x core.

2.4.6.1.3 OSD negotiation

In order to minimize the required console management software usage, our device will support auto-negotiation of the number of OSDs that are present on a given shared I/O port. The EEPROM will configure the allowable number of OSDs for each shared port and will be loaded as the default configuration of the system.

Once our switch completes link training when a new link is enabled (due to a plug-in card being added or just coming out of reset in the system), it will begin the process of figuring out what types of devices it is connected to on each port. A new procedure is defined to support this, using a new DLLP that is shown here:

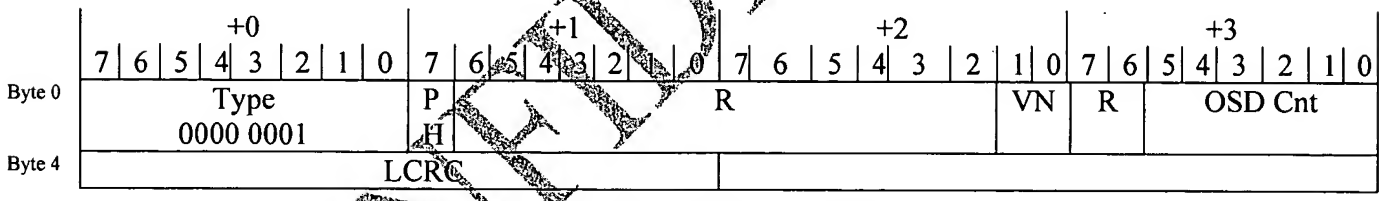


Figure 2.4-10: InitOSD DLLP format

Type – always set to 0000 0001 for an OSD negotiation DLLP
PH – Phase of OSD Negotiation (0 for InitOSD1 DLLPs, 1 for InitOSD2 DLLPs)
R – Reserved
VN – Version number (set to 00 for base OSD negotiation)
OSD Cnt – For InitOSD1 DLLPs, the number of OSDs present in the device… for InitOSD2 DLLPs, the negotiated number of OSDs for the link

The protocol very closely resembles the base specification's method of FC initialization. For the "shared I/O base mode" negotiation, the VN field must be set to 00. This means that the OSD and VC will explicitly be used for all wireline communication between two devices.

For "shared I/O extended mode", the VN field can be set to 01 which means. This mode means that the RN field will be used to explicitly map traffic onto buffer resources

between the link partners. This allow a larger number of OSDs to share a buffer if desired.

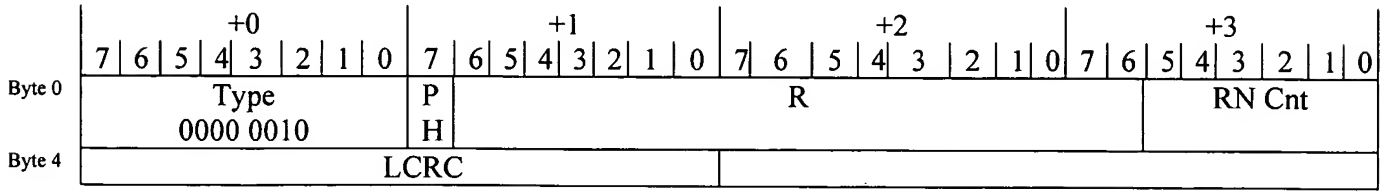
1. The OSD state machine will begin transmitting the same packet, an InitOSD1 DLLP, over and over every clock cycle until it receives an InitOSD1 DLLP from the link partner's OSD state machine. At that point, the device will check the value of the "OSD Cnt" field received from the link partner and compare that to the switch's number of OSDs (which was being advertised in the "OSD Cnt" field of the packets it initiates). The lesser of the two numbers will be the negotiated number of supported OSDs for that link. If the link partner never sends any InitOSD1 DLLPs after a timer expires (3 us), it is a non-shared port and our swich will proceed to normal flow control initialization.
2. At that point, the OSD state machine will continue sending continuous InitOSD1 DLLPs, but now it'll put the newly negotiated value in the "OSD Cnt" field. Once it receives a DLLP from the link partner with the same number in the "OSD Cnt" field, the state machine moves on to step 3.
3. The OSD state machine now transmits the same data except it sends it as an InitOSD2 DLLP by setting the PH bit in the DLLP. The fact that the state machine is now sending this type of DLLP means that it understood the OSD negotiation procedure from its link partner. A timer is also started at this point, and if the timer expires (3 us) before an InitOSD2 is received from the link partner, the state machine is reset and the process begins again. If the state machine receives an InitOSD2 from its link partner, OSD negotiation is complete and it stops transmitting InitOSD2 DLLPs.

2.4.6.1.4 Shared resource initialization

Once the number of OSDs is known on a link, shared resource initialization begins if the VN field was set to 01 during OSD negotiation. If the VN field was 00, this step is skipped.

This mechanism allows the two devices to map multiple OSD/VCs onto a common resource, or buffer, if desired. The results of the shared resource initialization will be stored in a register for software to read during system setup update. If any remapping of OSD/VCs to buffers is required, it can be done at that point.

This protocol performs the same basic steps as OSD negotiation. If a link partner does not respond with InitRN1 DLLPs within a 3 us time interval, it does not support shared resource initialization.



Type – always set to 0000 0010 for a RN initialization DLLP
PH – Phase of RN initialization (0 for InitRN1 DLLPs, 1 for InitRN2 DLLPs)
R – Reserved
RN Cnt – Total number of shared buffers that can be used by the link parnet

The step by step breakdown of the shared resource initialization protocol is shown below.

1. The RN state machine will begin transmitting the same packet, an InitRN1 DLLP, over and over every clock cycle until it receives an InitRN1 DLLP from the link partner's RN state machine.
2. The RN state machine now set the PH bit so that the DLLPs are now InitRN2 type. The fact that the RN state machine is now sending this type of DLLP means that it understood the RN initialization procedure from its link partner. A timer is also started at this point, and if the timer expires (3 us) before an InitRN2 is received from the link partner, the state machine is reset and the process begins again. If the state machine receives an InitRN2 from its link partner, OSD negotiation is complete and it stops transmitting InitRN2 DLLPs.

If this step is skipped because a link partner does not support it, the routing header will always have the RNP bit set to 0 since the RN field will not be used.

2.4.6.1.5 System setup update

At this point, the number of OSDs (and optionally number of buffer resources) available in the link partner is known by the hardware. Both values are written to a register so that software can see the results.

Based on the OSD negotiation that established the allowable OSDs on that port, the switch will write to the 16 buffer resource registers based on the results of OSD negotiation. These registers will contain some fields that specify what the encodings are (internal to the switch) that point to a particual OSD/VC. The EEPROM will have already set up default credit amounts for both the header and data buffers.

For example, if two OSDs were negotiated, the 16 registers might look something like this:

Buffer resource 0 Register: OSD=0, VC=0, valid=1
Buffer resource 1 Register: OSD=1, VC=0, valid=1
Buffer resource 2 Register: OSD=x, VC=x, valid=0
...

This means that ingressing TLPs on OSD0/VC0 will use buffer resource 0 space since the link partner will set the RN=0 when it sets the OSD=0 in the AS header. Incoming TLPs on OSD1/VC0 will have RN=1, OSD=1 in the ExAS header.

The hardware will now pause and query a control bit, halt_on_osd_complete, to determine what to do next. If that control bit is set (by the EEPROM during system

boot), the hardware will wait for software to clear it before proceeding to FC initialization. If the bit is cleared (ie, wasn't set by the EEPROM programming), the hardware will proceed to FC initialization immediately.

The purpose of this bit is to allow the system software/console management software an opportunity to take a look at what the results were of OSD negotiation. It can then go re-provision the buffer resources prior to FC initialization by reallocating space to different OSDs if necessary (and possibly VCs; this particular topic is covered in a sub-section below). Note that the hardware makes no assumptions on the "correctness" of this reprogramming and will make no attempt to recover from invalid programming at this point.

2.4.6.1.6 Flow Control (FC) initialization

If only 1 OSD is present as a result of OSD negotiation or OSD negotiation was skipped because the link partner is a non-shared device, the state machine will begin the normal PCI-Express base specification FC initialization procedure. However, if more than one OSD is present, the state machine will begin "shared I/O base FC initialization". If the shared resource initialization step was successful, the state machine will begin "shared I/O extended FC initialization." There is also another mechanism called *Buffer Retry Mode* that is not implemented and is explained in section 0.

2.4.6.1.7 Shared I/O Base FC Init

A new DLLP is created to convey the FC initialization information. This DLLP is shown here:

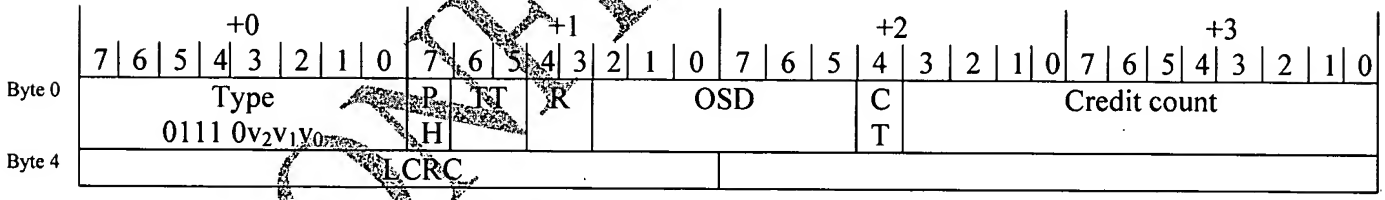


Figure 2.4-11: InitFC-H/InitFC-D DLLP format

Type – upper nibble set to 0111 for an FC initialization shared-link DLLP. The lower nibble specifies the VC number.
PH – Phase of FC Initialization (0 for InitFC1 DLLPs, 1 for InitFC2 DLLPs)
TT – transaction type (00 for Posted, 01 for Non-posted, 10 for Completions)
R – Reserved
OSD – OS Domain, ranging from 0 to 63 to specify the unique OSDs on the link
CT – Credit type (0 for header credits, 1 for data credits)... this basicially identifies the DLLP as either an InitFC1_H or an InitFC1_D
Credit count – contains either the 12-bit data credit count or the 8-bit (upper 4 bits are zeros) header credit count, based on the value of the CT bit

This section shows how shared-port FC initialization (where shared resource initialization was skipped) is performed. The pattern will closely resemble the base specification method of advertising normal FC. The state machine begins sending InitFC1_H and InitFC1_D DLLPs. It will send them in a repeating sequence as shown below as an example of a link that negotiated 2 OSDs. For this example, our switch was configured to only enable VC0 on each OSD:

InitFC1_H (OSD = 0, VC = 0, Posted, header)
InitFC1_D (OSD = 0, VC = 0, Posted, data)
InitFC1_H (OSD = 0, VC = 0, Non-posted, header)
InitFC1_D (OSD = 0, VC = 0, Non-posted, data)
InitFC1_H (OSD = 0, VC = 0, Completion, header)
InitFC1_D (OSD = 0, VC = 0, Completion, data)
InitFC1_H (OSD = 1, VC = 0, Posted, header)
InitFC1_D (OSD = 1, VC = 0, Posted, data)
InitFC1_H (OSD = 1, VC = 0, Non-posted, header)
InitFC1_D (OSD = 1, VC = 0, Non-posted, data)
InitFC1_H (OSD = 1, VC = 0, Completion, header)
InitFC1_D (OSD = 1, VC = 0, Completion, data)

So for this example, 12 unique DLLPs will calculate the credits for each OSD/VC enabled. Anytime more VCs are enabled using the normal mechanism for PCI-Express, this procedure is run in the same fashion.

Since VC0 should always be enabled, the switch should "expect" to receive the same 12 DLLPs from the link partner.

This pattern will continue until all corresponding DLLPs have been received from the link parter. At that time, the switch will begin sending the same sequence of DLLPs, except they will be InitFC2_H and InitFC2_D DLLPs (again, all 12 in a repeating sequence). Once the switch receives an InitFC2 DLLP from its link partner, FC initialization is complete. Note that whenever the FC1 phase is complete, TLPs can begin transmitting on the link. FC2 is just used to finally complete the handshake.

The EEPROM will contain the pre-calculated amount of credits to advertise and load these values into our internal registers at boot time. This can result in non-optimal buffer usage in the event the default provisioning is set up for a device that does not contain the same number of OSDs and/or VCs. The actual hardware defaults will assume all 16 OSDs are enabled, all with VC0. As such, the credits will be equally split across all 16 OSDs for all transaction types.

2.4.6.1.8 Shared I/O extended FC init

This section shows how shared-port FC initialization (where shared resource initialization was not skipped) is performed. The pattern will closely resemble the base specification method of advertising normal FC. The state machine begins sending InitFC1_H and InitFC1_D DLLPs. It will send them in a repeating sequence as shown below as an example of a link where we have 2 shared resources (RN = 2) as provisioned by the EEPROM:

InitFC1_H (RN = 0, Posted, header)
InitFC1_D (RN = 0, Posted, data)
InitFC1_H (RN = 0, Non-posted, header)
InitFC1_D (RN = 0, Non-posted, data)
InitFC1_H (RN = 0, Completion, header)
InitFC1_D (RN = 0, Completion, data)
InitFC1_H (RN = 1, Posted, header)
InitFC1_D (RN = 1, Posted, data)
InitFC1_H (RN = 1, Non-posted, header)
InitFC1_D (RN = 1, Non-posted, data)
InitFC1_H (RN = 1, Completion, header)
InitFC1_D (RN = 1, Completion, data)

2.4.6.1.9 Buffer Retry Mode

In Buffer Retry Mode, FC Initialization will be performed according to the way it is specified in the PCI Express Base Specification. If the port is shared, it will be transparent at this stage – FC Initialization is done only on a per VC basis. The partitioning of the buffer resources per OSD within each VC is determined during the configuration of the device-specific registers.

In Buffer Retry Mode, the inline data buffer is partitioned by VC as well as by OSD, and sets aside a "surplus" amount of memory that all types of packets can access regardless of OSD. The partitioning of the memory for Buffer Retry Mode is shown in Figure 12 and the variables to be programmed are shown in Table 4.

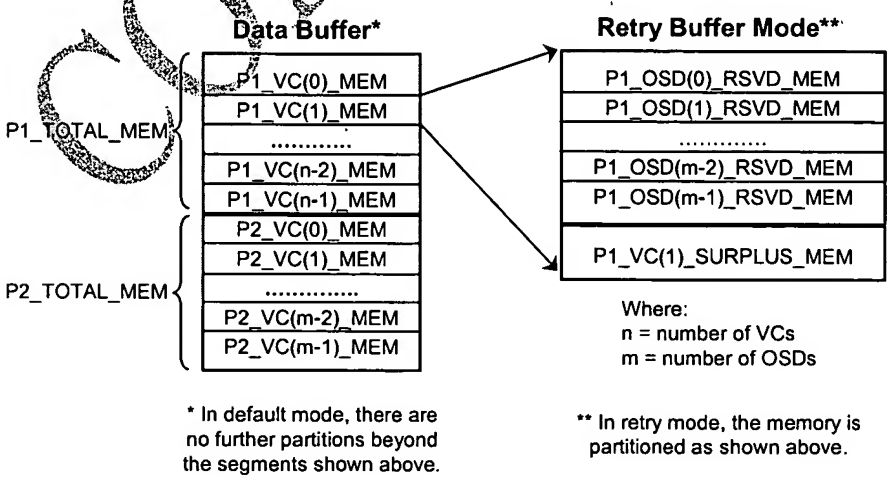


Figure 12: Breakdown of the Data Buffer

Since Flow Control DLLPs report credits solely by VC, and the Buffer Retry Mode breaks the buffer down even further into OSDs per VC, a packet could be transmitted that had enough flow control credit, but would still not get stored in the data buffer. This would happen if the packet belonged to an OSD that no longer had free space in its section of the data buffer, even though there appears to be credit according to flow control.

Instead of dropping packets that could not get stored in the data buffer due to lack of resources for a particular OSD, the Ack/Nak DLLPs are used to inform the other end of the link to retry the packet. According to the PCI Express Base Specification, an Ack/Nak DLLP is generated in the receive side of the Data Link Layer and is transmitted to the other side of the link to inform the transmit side of the Data Link Layer to either free its buffer resources that corresponds to the packet (if Ack'd because there were no data integrity errors), or retransmit the retry buffer (if Nak'd because there were data integrity errors). If repeated attempts to transmit a TLP are unsuccessful, the transmitter will instruct the Physical Layer to retrain the link.

The VMAC takes the Ack/Nak DLLPs to another level by allowing the Rx Transaction Layer Module to alter the DLLP depending on whether the TLP is able to get stored in the Data Buffer. If the TLP cannot be stored, the Ack DLLP that corresponds to the TLP is changed to a Nak DLLP and is transmitted to the other side of the link. One of the reserved bits in the Nak DLLP will also be set to differentiate between a Nak caused by a data integrity error (which can ultimately retrain the link) and a Nak that is caused by a buffer retry condition. This bit is shown in red in Figure 13 (Note: Using reserved bits is acceptable since non-zero values in Reserved fields are ignored and will not cause errors with other PCI Express links).

If the TLP has enough resources in the data buffer, but is stored in the surplus segment, one of the reserved bits in the Ack DLLP will be set to notify the other side that the OSD corresponding to that particular TLP is reaching buffer saturation. This bit will be used by the Transaction Arbiter in the Tx Presentation Module. The Transaction Arbiter will skip the Transaction Queue that corresponds to the particular VC/OSD group on the proximate turn. This will give the Rx data buffer some time to free up its resources.

Programmable Variables	Mode	Width	Description
P1_TOTAL_MEM	Default & Buffer Retry	00 01 10 11	32KB total memory allocated to Port 1 64KB total memory allocated to Port 1 96KB total memory allocated to Port 1 128KB total memory allocated to Port 1
P1_VC(n)_MEM	Default & Buffer Retry	6 bits	0x0 = 128KB, 0x1 = 2KB, 0x2 = 4KB, …, 0x3F = 126KB where

P1_VC(n)_SURPLUS_MEM	Buffer Retry Only	3 bits	0x0 = 16KB, 0x1 = 2KB, 0x2 = 4KB, …, 0x7 = 14KB where P1_VC(n)_SURPLUS_MEM < P1_VC(n)_MEM
P1_OSD(m)_RSVD_MEM	Buffer Retry Only	6 bits	0x0 = 128KB, 0x1 = 2KB, 0x2 = 4KB, …, 0x3F = 126KB where
P2_TOTAL_MEM	Default & Buffer Retry	N/A	P2_TOTAL_MEM = 128KB - P1_TOTAL_MEM
P2_VC(n)_MEM	Default & Buffer Retry	6 bits	0x0 = 128KB, 0x1 = 2KB, 0x2 = 4KB, …, 0x3F = 126KB where
P2_VC(n)_SURPLUS_MEM	Buffer Retry Only	3 bits	0x0 = 16KB, 0x1 = 2KB, 0x2 = 4KB, …, 0x7 = 14KB where P2_VC(n)_SURPLUS_MEM < P2_VC(n)_MEM
P2_OSD(m)_RSVD_MEM	Buffer Retry Only	6 bits	0x0 = 128KB, 0x1 = 2KB, 0x2 = 4KB, …, 0x3F = 126KB where

Table 4: Programmable Variables for Data Buffer

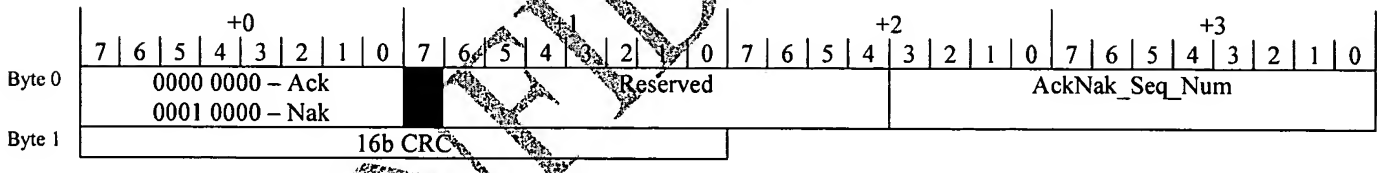


Figure 13 : DLLP Format for Ack/Nak Packets

2.4.6.1.10 Pseudo-device discovery

Once the FC init is done, the devices are ready to exchange TLPs, so device discovery can proceed. This is another optional checkpoint at which we could implement another halt bit, halt_after_fc_init. If the I2C set this bit at boot time, the hardware will now pause again. Console management software can now come in using the I2C bus and use the COP to emulate device discovery by acting like a Root Complex. It could send type0 and type1 config cycles to the newly-plugged-in I/O device and figure out what VCs, buffers, and OSDs that device had available. It would load this information into local memory and essentially restart the whole process again for that I/O device.

This is an advanced feature that allows software the hooks it would need to set up optimal resources across VCs and buffers before real device discovery happens from the OS. So the steps would be as follows:

1. I2C initialization
2. Link training
3. OSD negotiation
4. FC initialization
5. "Pseudo" Device Discovery
6. Load results into local memory of console processor
7. Link training
8. OSD negotiation
9. System setup update (use newly calculated optimal buffer settings)
10. FC initialization
11. OS Device Discovery

2.4.6.1.11 Discovery

The discovery mechanism for shared devices uses the standard PCI-Express mechanism for discovery per OS domain. This mechanism sends Type0 and Type1 configuration cycles to determine which devices, if any, are present behind a link. Each one of these cycles is expected within the switch for each active OS domain.

A root complex will begin sending Type0 CFG cycles to its southbound PCI-Express port. It will discover the switch port, directly connected to the switch. It will discover the switch as a PCI-bridge, and no other devices on that link. It will then initiate Type1 CFG cycles to discover the devices behind that PCI Bridge on that port. Once inside the switch, the Type1 CFG cycles will discover all ports and PCI Bridges that have been assigned to that OS domain. A shared port will appear as assigned to that OS domain.

When a root complex discovers a shared port, it has no knowledge that the port is shared. As such, the port will appear to that root complex as a PCI-Bridge, the same as all other ports. In the initial switch implementation, there will be 16 OS domains possible. Each root complex will be mapped to one of those OS domains. This mapping will have a specific AS turn pool encoding as a 16 port switch. From the switch view, it will always have a 16 port switch as its link partner.

For CFG cycles sent to a shared link, the CFG cycle will be encapsulated within the AS PEI8. This will be sent with the turn pool encoding assigned to the appropriate OS domain. The response to that CFG cycle will depend on the number of real OS domains supported by that link partner. For a given shared link partner, it will support a certain number of OS domains. The shared link partner will only respond to CFG cycles which are mapped to OS domains that it supports.

In the following diagram, there is a 4 OS domain shared controller tied to the switch. The switch supports 16 OS domains, enumerated as a 16 port virtual AS switch. The

Shared controller supports 4 OS domains, enumerated as a 4 port virtual AS switch. The switch always sends packets encapsulated to a 16 port virtual AS switch. The controller always sends packets encapsulated to a 4 port virtual AS switch. If a port ever sees packets encapsulated with turn pool beyond the range it supports, the packets are no responded to at the transaction layer.

This method allows standard PCI discovery to work. In the example below, each OS domain maps to a given virtual AS switch port. Root Complex 1 initiates a Type1 CFG cycle destined for the shared controller. The switch changes the Type1 to a Type0 cycle at Port10 prior to sending the CFG cycle to the controller. This CFG cycle is encapsulated to a single virtual AS port, and the controller responds by sending the response on the corresponding return virtual AS port.

If a Root complex was to exist that was mapped to a virtual AS port that the controller does not have, e.g. virtual AS port 10, the controller would drop the packet. At the transport level, this would be the equivalent of a timeout on the CFG read, and the Root complex would assume no I/O device is present on that logical PCI bus. This mechanism allows all devices to be discovered in the same method, with no I/O device required to know the full fabric topology.

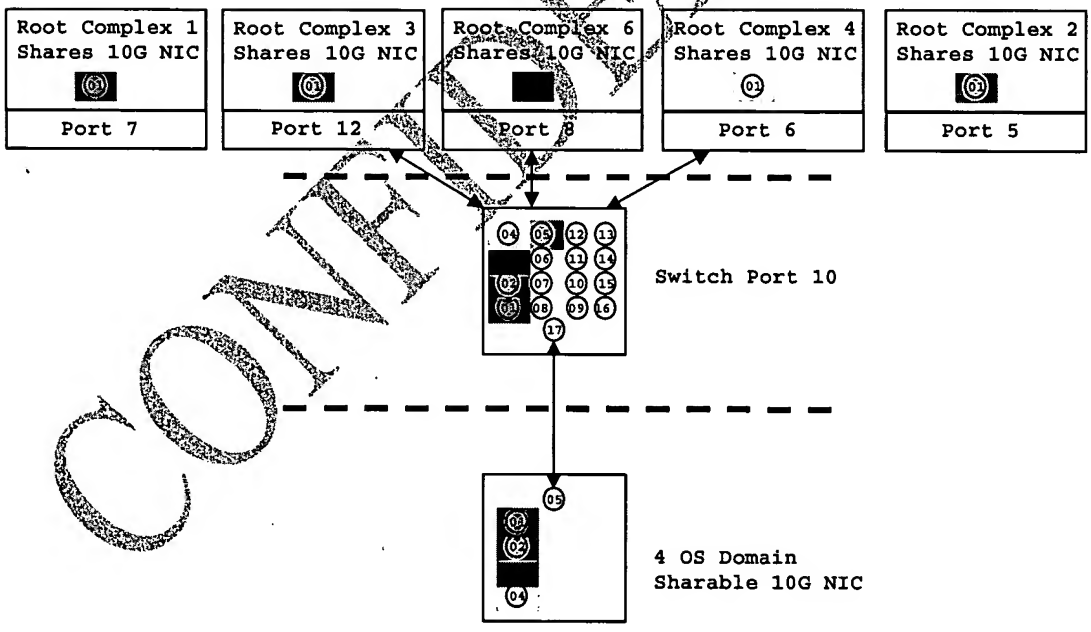


Figure 2.4-14: Example of a Shared Switch

2.5 Error Handling

Two aspects of a PCI Express switch make its error handling behavior much different than that of a PCI Bridge:

- First, transient link errors can typically be corrected automatically by hardware in the Data Link Layer. This eliminates the need to report these errors by setting any bits in the PCI status registers and eliminates any problems produced when the failed packet was being forwarded on behalf of another device. This is because the packet will ultimately be transmitted correctly, so no error handling procedures beyond the scope of a single link need to be considered. Repeated failures will ultimately cause a fatal error to be recorded and the faulty link will be shut down.
- Second, non-posted transactions are managed with a completion message that eliminates the need for a forwarding agent to keep track of expected response messages. So a switch can blindly forward transactions without determining the message type or worrying about mapping errors back to the originating master.

When a Data Link Layer or Physical Layer error which cannot be associated with a particular packet or OSD, the error is logged separately for each OSD and error messages if any are sent to each port sharing the port with an error.

The Nexis Switch implements the PCI Express Advanced Error Reporting Capability registers in addition to the PCI Express basis error reporting registers and the legacy PCI mapped error registers. The Advanced Error Reporting Capability registers provide detailed error logging, error masking and error severity control registers. It also provided a header logging register for the first unmasked error that's logged. Refer to the *Configuration Registers* section for detailed description of the Advanced Error Reporting Capability registers.

2.5.1 Error Types

PCI Express errors are classified as Correctable and Uncorrectable errors. Uncorrectable errors are further classified as Fatal or Non-Fatal errors. Errors are also classified based on the source of the error as Transaction Layer Errors, Data Link Layer Errors and Physical Layer Errors. The following sections describe each of the errors and how they are handling in the Nexis Switch.

2.5.1.1 Correctable Errors

Correctable errors are those which are localized to a single PCI Express link and can be automatically corrected by hardware. All correctable errors are automatically corrected by a retransmission of the faulty packet. An ERR_COR message reporting the occurrence of the error may optionally be sent to the root complex. The message is sent only if the error is not masked and the SERR Enable bit is set in the Command Register and Bridge Control register.

2.5.1.1.1 Physical Layer Errors

- **Receiver Error**

Physical Layer receivers may optionally check for errors and report them by sending an ERR_COR message to the root complex. Any DLLP or TLP being received that is in error should be discarded and any storage allocated made available. The error will be automatically corrected when a NAK DLLP is received and the packet is re-transmitted.

The Physical Layer will check for disparity errors and invalid symbols. If any of these errors occur, the Physical Layer will report it. If the error (either one) is part of a valid TLP, the Rx Link Layer will send a NAK DLLP for the corresponding TLP and will not report an error since it was already done by the Physical Layer. If the error is detected in the Data Link Layer in time (within 3 clock cycles from the start of packet), the packet will be purged and the Rx Transaction Layer will never see the packet. If the error is not detected in time, the TLP will be forwarded to the Transaction Layer with an error indication at the end of the packet.

2.5.1.1.2 Data Link Layer Errors

- **Bad TLP**
  This error is set when the link layer detects a packet with one of the following.
  - Bad CRC
  - Incorrectly nullified packet (TLP ends with EDB, but the LCRC is not inverted)
  - Incorrect packet sequence number (not duplicate).
- **Bad DLLP**

This error occurs when a CRC check fails on a DLLP.

- **Replay Timer Timeout**

This error occurs when the REPLAY_TIMER has been exceeded by a given TLP, which occurs when no ACK or NAK DLLP is received within that time period. This error is automatically corrected by NAK'ing the TLP and forcing a re-transmission.

- **REPLAY_NUM Rollover**

This error occurs when a given TLP was unsuccessfully retransmitted REPLAY_NUM times. This condition is automatically corrected by signaling the Physical Layer to retrain the link. Once retraining is successful, the TLP can again be retransmitted (and REPLAY_NUM is reset).

2.5.1.2 Uncorrectable Errors

Uncorrectable Errors are those which disrupt the functionality of the PCI Express port but cannot be corrected by hardware. Using the Advanced Error Reporting Capability Registers each uncorrectable error can be configured to be sent with an ERR_FATAL or ERR_NONFATAL message to the root complex or can be masked off from sending a message. The error messages are sent only if the SERR Enable bit is sent in both the Command Register and Bridge Control Register.

2.5.1.2.1 Physical Layer Errors

- **Training Error**
  This error occurs when a device fails to establish a link with its partner. An error message is sent to the root complex if enabled, and the link is taken down.

2.5.1.2.2 Data Link Layer Errors

- **Data Link Layer Protocol Error**
  This error is caused when the TX MAC receives an ACK/NAK DLLP with a sequence number that does not correspond to any of the packets in the retry buffer.

2.5.1.2.3 Transaction Layer Errors

- **Unsupported Request**

An Unsupported Request Error is generated for the following conditions.

  - A request cannot be mapped to any address space mapped within the device or to any egress ports.
  - Downstream port of the switch receives a configuration request with Device number 1-31. The port will terminate the transaction and not issue it on the link.
  - A packet if forwarded to an egress port to be transmitted across the link, but the link is in DL_Down state.

- **ECRC Error**

Logging this error is optional. The Nexsis switch will not verify the ECRC for forwarded packets. All transactions originating from the switch (COP) will not contain ECRC. All transactions destined to the switch (Type1/Type0 headers and Device specific registers) that have ECRC will not be checked. The ECRC Generation Capable and ECRC Check Capable bit in the Advanced Error Capability register is hardwired to zero.

- **Malformed TLP**
  The receiver of TLP packets generates this event when an inconsistency in the formation of a packet is detected at the receiver (destination). There are several conditions that require detection and reporting, some others are optional. The table below shows the conditions that are supported and not supported by the switch.

Malformed Packet Errors	Supported?

Data payload exceeds max payload size	Yes
Actual data length does not match data length specified in the header	Yes
Start memory DW address crossing a 4KB boundary	No
TD field = 1 but no ECRC	No
Byte enable violation detected	Yes, only for writes to COP
Packets with undefined type field	Yes
Multiple completions with read data that violate RCB	No
Completions with a configuration request retry status in response to a request other than a configuration request	No (is this optional?)
TC contains a value not assigned to an enabled VC within the TC/VC mapping for the receiving device	Yes
Transaction type requiring use of TC0 has TC value other than zero.	No (maybe V2)
Routing is incorrect for transaction type (e.g. transactions requiring routing to RC detected moving away from RC)	No (maybe V2) (this is an ALM check)
Msg/MsgD messages with 000b routing received at upstream port	?? (ALM check)
Msg/MsgD messages with 011b routing received at downstream port	?? (ALM check)

A malformed TLP will be discarded and an ERR_NONFATAL or ERR_FATAL message may be sent to the root complex. No Nak DLLP is sent in response to a Malformed TLP, and the flow control credits are not updated. A Completion Response will not be issued for non-posted transactions with a malformed TLP.

- **Receiver Overflow**
  A Receiver may optionally check for Receiver Overflow errors (TLPs exceeding CREDITS_ALLOCATED). If this condition is detected, TLP(s) are discarded without modifying the CREDITS_RECEIVED and any resources that had been allocated for the TLP(s) are de-allocated. (Not supported right now in the FPGA)
- **Completion Timeout**
  When a non-posted transaction fails to return a completion message within the subscribed time limit, then a *completion timeout* error has occurred. The Nexsis switch will not master any non-posted transactions, and so will never generate a *Completion Timeout* error for any packets going off-chip.

- **Completer Abort**
  This error occurs when the Completer of a request is unable to process the request due to a component-specific error condition. There are no known conditions for which the switch will generate this error.

- **Unexpected Completion**
  This error occurs when a completion message is received that cannot be matched to any outstanding requests. The switch will report this error only for the completion targeted to the switch as an endpoint. The ALM detects this condition and notifies the MAC to log the error.

- **Flow Control (FC) Protocol Error**

  The receiver of Flow Control DLLP packets generates this event when a violation of the flow control protocol is detected at the receiver (destination). There are several conditions that require detection and reporting, some others are optional. The following conditions may be checked and violations reported.
  • During FC initialization for any Virtual Channel the VC must advertise credits equal to or greater than the minimum for that FC type

  • IF an Infinite Credit advertisement (value of 00h) has been made during initialization, THEN any future update credit values must be set to zero.

- **Poisoned TLP**
  A poisoned TLP is one where the EP field if the header is set indicating that the TLP is known to contain an error. If the packet is already poisoned the switch will not issue an error message unless it is the final target of the poisoned TLP.

  If the error is an uncorrectable ECC error in the internal data buffers of the switch for one of the transit packets the switch sets the EP bit in the header, logs the error and forwards the packet.

  When a switch forwards a poisoned TLP, the receiving side must set its Detected Parity Error bit, and the transmitting side must set its Master Data Parity Error bit if the Parity Error Response bit in the Bridge Control register is set.

2.5.1.2.4 Cut-Through Error Handling

If the cut_thru_enbl bit is asserted in the Nexsis Switch may, a TLP could be forwarded from the ingress port to the egress port before it has been completely received on an ingress port. This complicates the error handling in the conditions where the TLP would otherwise be discarded. If an error does occur on a cut-through packet after it has begun transmission out an egress port, then the TLP must be 'nullified' to indicate to the receiving device that an error has occurred. A TLP is nullified by either using the inverted value for its LCRC or by signaling the physical layer that it must use an EDB

symbol instead of an END symbol as the final framing symbol. The ingress port returns a NAK DLLP to the TLP source and the egress port purges the packet from the Replay buffer. When the endpoint finally receives the TLP and detects the EDB symbol and the inverted CRC, it purges the packet and does not return a NAK DLLP.

2.5.2 PCI 2.3 Error Reporting

All PCI Express ports will update the error status bits in the PCI 2.3 configuration space as appropriate in order to maintain compatibility with legacy drivers. Note that these status bits are independent of any status maintained in the Advanced Error Reporting Status or control registers. In particular, setting or clearing an Advanced Error Reporting Status bit should not clear the corresponding bit in the PCI 2.3 configuration registers – these must be left to be explicitly managed by software and is performed by the COP.

Each PCI Express port connects implicitly to a PCI-PCI virtual bridge. Each of these bridges will implement a complete and independent PCI-PCI bridge type 1 configuration space.

Note that the primary bus of each bridge is the one closest to the Root Complex, and that this can vary depending on how the Nexsis Switch is installed in a system.

The following sections detail how the Nexsis Switch should report errors to remain compatible with legacy PCI 2.3 software.

2.5.2.1 Primary side of P2P Bridge

- **Detected Parity Error**

  This error will be set when ever the primary side of the internal P2P bridge receives a poisoned TLP. In the Nexsis switch the RX ingress MAC (Root Port) would set the Detected Parity Error bit in the Primary Status Register when receiving a poisoned TLP.

- **Signaled System Error**

  The TX MAC of the upstream port must set the Signaled System Error if an uncorrectable error message (ERR_FATAL or ERR_NONFATAL) is transmitted.

- **Received Master Abort**

  This error needs to be detected only by PCI Express device which originally initiates a transaction and hence not applicable to the Nexsis switch.

- **Received Target Abort**

  This error needs to be detected only by PCI Express device which originally initiates a transaction and hence not applicable to the Nexsis switch. Signaled Target Abort – hardwired to 0 since we will not abort any transactions as an endpoint.

- **Master Data Parity Error**

  This error is detected when forwarding a Poisoned TLP from the secondary side of the bridge to the primary side. In the Nexsis switch the TX MAC of the upstream port must set the Master Data Parity Error bit in the Primary Status register if the Parity Error Response bit in the Command Register is set.

2.5.2.2 Secondary side of P2P Bridge

- **Detected Parity Error**

  This bit is set when the secondary side of P2P bridge receives a poisoned TLP. In the nexis switch the Rx MAC of the downstream port must set the Detected Parity Error bit in the Secondary Status register when it receives a poisoned TLP.

- **Received System Error**

  The Rx MAC of a downstream port must set the Received System Error if an uncorrectable error message (ERR_FATAL or ERR_NONFATAL) is received.

  Received Master Abort

  This error needs to be detected only by PCI Express device which originally initiates a transaction and hence not applicable to the Nexsis switch.

- **Received Target Abort**

  This error needs to be detected only by PCI Express device which originally initiates a transaction and hence not applicable to the Nexsis switch.

- **Signaled Target Abort**

  There are no known conditions for which the switch should Target Abort a transaction targeted to it.

- **Master Data Parity Error**

  This error is detected when forwarding a poisoned TLP from Primary to Secondary. In the Nexis switch the TX downstream MAC would set the Master Data Parity Error bit in the Secondary Status register when transmitting a poisoned TLP.

2.5.3 Error Reporting

Errors may be reported by either generating an explicit error message, or through the Completion Status field in a Completion header. A completion response is used for reporting errors resulting from a non-posted request, while explicit error messages are used for all other types of messages. Note that a Completion Status may only be used by the intended target of the originating message. Thus, a non-posted message that is being routed through the switch will never have a Completion generated by the switch, and so all errors reported for that message will use explicit messages. The only messages that will use the Completion response to report errors will be those messages that are targeted to internal registers of the switch itself.

Note also that only Unsupported Request and Completer Abort errors are reported in a completion response. All other errors, even for non-posted requests, will generate an explicit error message.

2.5.3.1.1 Completion Status Response

The format of a completion header used to respond to an error condition is as follows:

+0	+1	+2	+3

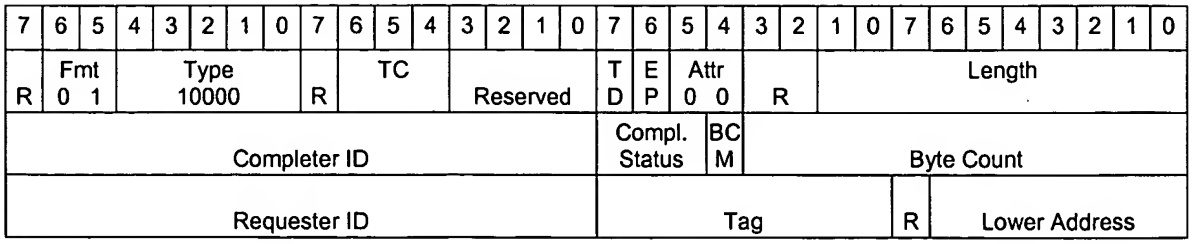


Figure 2.5-1: Format of Completion Header

The following table shows the fields and their values for completion headers reporting error conditions:

Field	bits	Description	Value	Variable
Length	9:0	Always zero for error completions	0	No
R	11:10	reserved	0	No
Attr	13:12	Copied from request header		Yes
EP	14	Indicates TLP is poisoned	0	No
TD	15	Indicates presence of TLP digest	0	No
Reserved	19:16	reserved	0	No
TC	22:20	Copied from request header		Yes
R	23	reserved	0	No
Type	28:24	Indicates Msg type	10000	No
Fmt	30:29	Indicates 4 DW header, no data	0b01	No
Byte Count	43:32	The remaining byte count for request		Yes
BCM	44		0	No
Compl. Status	47:45	000 = Successful Completion 001 = Unsupported Request 010 = Configuration Request Retry Status 100 = Completer Abort		Yes
Completer ID	63:48	Bus #, device # and function # of unit generating completion (and reporting error)		Yes
Lower Address	70:64	Unused	0	No
R	71	reserved	0	No
Tag	79:72	Copied from request header		Yes
Requester	95:80	Copied from request header		Yes

ID				

Table 2.5-1 Completion Header Fields

2.5.3.1.2 Explicit Error Messages

Explicit error messages are generated for every correctable or uncorrectable error which is not masked in the Advanced Error Capabilities register. For shared ports an error message for an error from a port which cannot be associated specifically to an OSD (for example, Training Error or Reply Timer Timeout) should be sent to all the RCs sharing the downstream switch port.

Error messages generated are one of the following three types:

Type	Description
ERR_COR	Issued when component or device detects a correctable error on the PCI Express interface
ERR_NONFATAL	Issued when the component or device detects a Non-fatal, uncorrectable error on the PCI Express interface
ERR_FATAL	Issued when the component or device detects a Fatal, uncorrectable error on the PCI Express interface

Table 2.5-2: PCI Express Error Messages

The format of all error messages is shown in the following table:

+0								+1								+2								+3							
7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
R	Fmt 0 1			Type 10000				R	TC 0			Reserved				T D	E P	Attr 0 0		R				Length 0							
Requester ID																Tag0								Message Code							

Figure 2.5-2: Format of Error Messages

All error messages are 16-bytes in length, with the following fields and values:

Field	bits	Description	Value	Variable
Length	9:0	Unused – always zero	0	No
R	11:10	reserved	0	No
Attr	13:12	Attributes – always zero	0b00	No
EP	14	Indicates TLP is poisoned	0	No

TD	15	Indicates presence of TLP digest	0	No
Reserved	19:16	reserved	0	No
TC	22:20	Traffic class, must be zero	0	No
R	23	reserved	0	No
Type	28:24	Indicates 'Msg' type	10000	No
Fmt	30:29	Indicates 4 DW header, no data	0b01	No
R	31	reserved	0	No
Message Code	39:32	0011 0000 = ERR_COR 0011 0001 = ERR_NONFATAL 0011 0011 = ERR_FATAL		Yes
Tag	47:40	Unused (no completion required)	0	No
Requester ID	63:48	Bus #, device # and function # of unit reporting error		Yes

Table 2.5-3 Error Message Fields

The spec requires all error messages to use a 4 DW header. This Error Message field is taken from the Error Log Register where the header of the first packet in error is stored.

2.5.3.1.3 Message Routing

The three lsb's of the message type field indicate the message routing mechanism, which is always '000' for error messages - indicating message should be routed to the Root Complex.

2.5.4 Header Logging

As part of the Advanced Error Reporting capabilities the Nexsis switch logs the TLP header for the first uncorrectable transaction layer error reported. Headers are logged only if the mask bit for the corresponding error is not set in Uncorrectable Error Mask register and the Uncorrectable Error Status bit pointed to by the First Error pointer is not set. Headers are logged in a 4 DWORD register for the following errors.

- Poisoned TLP received
- ECRC Check Failed (error is not supported)
- Unsupported Request
- Completion Abort (error is not supported)
- Unexpected Completion
- Malformed TLP

There are no variations in header logging logic for shared or non shared port.

2.5.5 Error Tables

The following table lists the errors that may occur and be detected on a single PCI Express link.

Layer	Error Name	Default Severity	Detecting Agent Action
P	Receiver Error	Correctable	Send ERR_COR to Root Complex (RC)
	Training Error	Uncorrectable (Fatal)	Send ERR_FATAL to RC
D	Bad TLP	Correctable	Send ERR_COR to RC
	Bad DLLP		Send ERR_COR to RC
	Replay Timeout		Send ERR_COR to RC
	REPLAY NUM Rollover		Send ERR_COR to RC
	Data Link Layer Protocol Error	Uncorrectable (Fatal)	Send ERR_FATAL to RC
T	Poisoned TLP Received	Uncorrectable (Non-Fatal)	Send ERR_NONFATAL to RC Log header of TLP
	ECRC Check Failed		Send ERR_NONFATAL to RC Log header of TLP
	Unsupported Request (UR)		Send ERR_NONFATAL to RC Log header of TLP
	Completion Timeout		Send ERR_NONFATAL to RC
	Completer Abort		Send ERR_NONFATAL to RC
	Unexpected Completion		Send ERR_NONFATAL to RC Log header of Completion
	Receiver Overflow	Uncorrectable (Fatal)	Send ERR_FATAL to RC
	Flow Control Protocol Error		Send ERR_FATAL to RC
	Malformed TLP		Send ERR_FATAL to RC Log header of TLP

Table 2.5-4: PCI Express Link Errors

Occurrence of any of the above correctable errors can be flagged in the Correctable Error Status Register and masked in the Correctable Error Mask Register.

Occurrence of any of the above uncorrectable errors can be flagged in the Uncorrectable Error Status Register and masked in the Uncorrectable Error Mask Register. Additionally, the uncorrectable errors can be programmed to be reported as either a fatal or non-fatal error by use of the Uncorrectable Error Severity Register. These registers are replicated for each OSD.

2.5.5.1.1 Error Signaling and Logging

Legend:

Type: C=correctable, NF=non-fatal, F=fatal – indicates type of error message that is generated

S= Supported N=Not Supported

Italicized errors correspond to specific error bit set

2.5.5.1.2 Rx Physical Layer Errors

Error	S/N	Packet Behavior	Reported Error	Type
Invalid symbol or disparity error	S	Discard pkt if error is part of pkt. Schedule Nak if TLP	*Rx Error* Rx Port Error reported by PLM (does not report if it was part of a pkt or not)	C
Link Error	S		*Receiver Error*	C
Training Error	S	N/A	*Training Error*	F

Table 2.5-5: Rx Physical Layer Errors

2.5.5.1.3 Rx Data Link Layer Errors

Table 2.5-6 Rx Data Link Layer Errors

Error	S/N	Packet Behavior	Reported Error	Type
Invalid sequence number on ACK or NAK DLLP	S	Discard DLLP	*DLL Protocol Error*	F
Duplicate Seq. Number on TLP	S	Discard TLP, schedule ACK	No error	
Unexpected Seq. Number on TLP	S	Discard TLP, send Nak	*Bad TLP*	C
LCRC error on TLP	S	Discard TLP (unless it's cut-through). Schedule Nak DLLP.	*Bad TLP*	C
LCRC error on DLLP	S	Discard DLLP	*Bad DLLP*	C
DLLP w/ unsupported type encodings	S	Discard DLLP	No associated Error	
FC Init protocol violations	S/N?	N/A	*DLL Protocol Error*	F
Rx Framing Violations	S	Discard pkt, send Nak	*Bad TLP*	C
TLP with EDB and inverted LCRC	S/N?	Discard pkt. No Nak scheduled	No associated Error	
Nullified TLP without EDB	S/N?	Discard TLP	*Receiver Error*	C

Table 2.5-7

2.5.5.1.4 Rx Transaction Layer Errors

Error	Opt	Packet Behavior	Reported Error	Type

Unmapped address	S	Discard TLP	*Unsupported Request*	F
FC overflow	N	Discard pkt. No Nak, No FC update.	*Receiver Overflow*	F
Malformed TLP [1]	S	Discard TLP, No Nak, No FC update	*Malformed TLP*	F
Pkt_length field < actual pkt length	S	Truncate pkt (or discard if S&F).	*Malformed TLP*	F
Pkt_length field > actual pkt length	S	Stop at END. Discard if S&F. No Nak, No FC update	*Malformed TLP*	F
Unexpected Completion [2]	S	Discard completion	*Unexpected Completion*	NF
Request violates programming model of Rx device	N	Discard request	*Completer Abort*	NF
Rx device unable to process request due to device-specific error condition	N	Discard request	*Completer Abort*	NF
Advertising more than 2048 credits for payload and 128 for header	S/N?		*Flow Control Protocol Error*	F
Did not advertise FC credit values >= min defined in Table 2-27 of PCI Express spec	S/N?		*Flow Control Protocol Error*	F
Non-zero credit values received after infinite credit advertised	S/N?		*Flow Control Protocol Error*	F
Link in DL_Down status	S/N?	Return completion as Unsupported Request, discard request	*Unsupported Request*	F
Receive poisoned TLP	S	Pass TLP thru, unless directed to switch, then discard (and return UR for non-posted requests)	*Poisoned TLP Received*	NF

[1] See conditions for malformed TLPs under section □ **Malformed TLP** on page 47.
[2] Nexsis Switch will never expect completions, so can just ignore them and let them pass thru switch.

Table 2.5-8 Rx Transaction Layer Errors

2.5.5.1.5 Tx Data Link Layer Errors

Error	S/N	Packet Behavior	Reported Error	Type
REPLAY_NUM rolls over	S	Retrain the link	*REPLAY_NUM Rollover*	C
REPLAY_TIMER expires	S	Retry entire buffer	*Replay Timeout*	C

Table 2.5-9 Tx Data Link Layer Errors

2.5.5.1.6 Tx Transaction Layer Errors

Error	Opt	Packet Behavior	Reported Error	Type
Invalid TC, OSD or type		Nak the transaction coming from the Switch Core.	None.	
TLP length exceeds Max_Payload_Size		Discard TLP (for TLPs with data payload only)		
Actual packet length is greater than pkt_length field.		Truncate and nullify packet.	Malformed TLP	
Actual packet length is less than pkt_length field		Nullify packet.	Malformed TLP	
Completion Timeout [1]		Discard corresponding request	*Completion Timeout*	NF

[1] The Nexsis Switch will not issue any Requests requiring Completions, so no timeout should ever be enabled

Table 2.5-10 Tx Transaction Layer Errors

2.6 QOS

There are two basic components to Quality of Service in our switch. First, the Traffic Class (TC) field in the transaction allows the driver to differentiate certain transaction flows and have them be mapped into a Virtual Channels (VC). So a particular message might be labeled with a TC of 7 for high-priority, while standard memory reads and writes might be labeled with a TC of 0. Our switch supports all 8 VCs. Second, our switch allows different OS Domains to share an I/O port, and each OSD will automatically receive its fair share of the bandwidth when that port is congested..

2.6.1 TC/VC Mapping

The PCI Express spec lists a 3-bit Traffic Class (TC) field that is present in the transaction header. This field is used to differentiate traffic so that it can be prioritized,

queued, and scheduled independently inside the various PCI Express devices. Although the spec says that anything other than TC0 is optional, our switch will accommodate all 8 traffic classes.

TC's are mapped onto Virtual Channels (VCs) in a vendor-defined manner according to the PCI Express spec. For our switch, each incoming TLP will have its TC field mapped into a VC based on software configuration settings. TC0 must always be mapped to VC0, but all other TCs may be mapped to VCs without restriction individually on a per-port basis.

Each RX MAC will have a TC mapping table that's just a flat-mapped lookup table that turns an OSD and TC into a Source QID. The Source QID is needed to know which of the 16 flow control resources were charged with the credits of the incoming TLP.

Once the destination port and destination QID are returned from the address_lookup_module, the transaction can actually be queued up now that all the data for the transaction is known.

2.6.2 Arbitration points

There are three arbitration points inside our switch that will be supported, two in the transaction_scheduler and one in the data_mover. In the transaction_scheduler, there are 17 sets of arbiters that run independently, one set per output port. Each set of arbiters will ensure each input port is allowed a fair share of the output port's bandwidth, and each OSD is allowed its fair share as well. All three levels of arbitration will be discussed in the next few sections.

2.6.2.1.1 Port arbitration (transaction_scheduler)

Port arbitration is the first step run within each port_arbiter at an output port (shown as ARB1 in the diagram on the next page). This level of arbitration will use a simple RR scheme to make sure no port is starved and the bandwidths are balanced. This RR scheme is fixed in hardware, no WRR is supported.

2.6.2.1.2 OSD/VC arbitration (transaction_scheduler)

Since there are 16 output buffer groups for each set of arbiters, a second level of RR arbitration (ARB2) to pick which transaction will be selected as the next one to be transmitted on a given output port. This RR scheme is fixed in hardware, no WRR is supported.

2.6.2.1.3 Input arbitration (data_mover)

The data_mover is the final stage (ARB3) that selects which input port is actually allowed to transfer data to each output port. Another level of RR is run to make sure each input port is serviced when more than one input port has data to send to a given output.

Note that the data_mover will skip over the "best" choice from the transaction_scheduler if a different input port that is idle is able to move its data instead. In this case, a skip

flag will be set such that the data_mover will not continue skipping the "best" choice more than once. Without this skip flag, one output could get very unlucky and get skipped over and over, stalling a transaction potentially indefinitely under certain traffic loads.

The following picture shows these levels of arbitration.

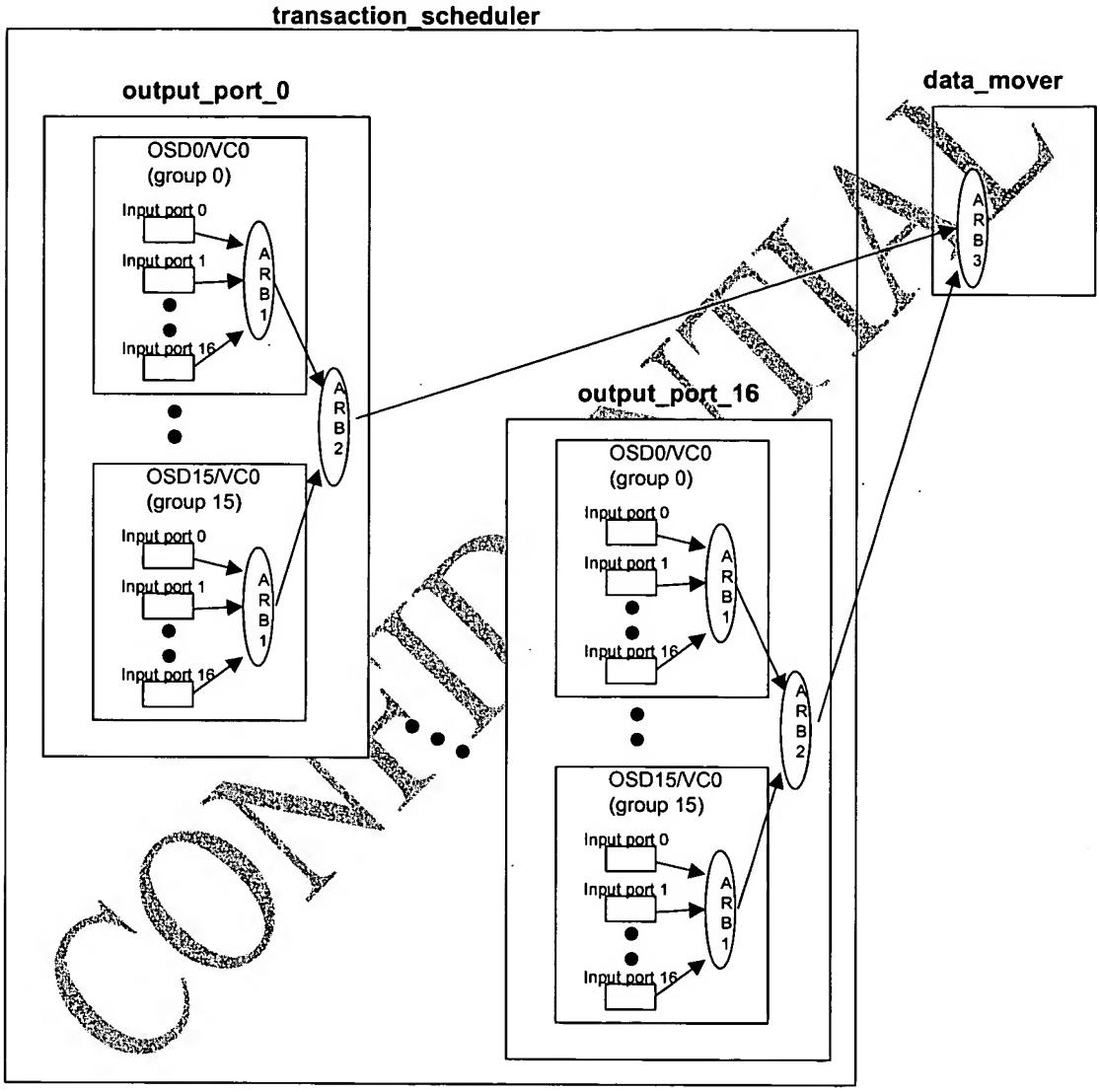


Figure 2.6-1: Transaction Scheduler Block Diagram

2.7 System Level Management Examples

2.7.2 Hot Plug Add of I/O Device

Our device will integrate a hot-plug controller, allowing system software and/or chassis management software access to several registers to control hot-plug functions. These registers are implemented in the PCI-Express capabilities structure and can be used to turn on and off LEDs, exchange information with the Power Controller in the chassis, and report the status of the various slots in our system.

Once a link has been trained, OSDs have been negotiated (if necessary), and FC initialization is complete, the hot plug process can begin.

For hot insertion, this process will start with the user pressing the Attention Button which will set a register bit in our switch for those P2P bridge headers. The COP will send an MSI up to the hot plug services software running on each OS for each OSD that share the newly-plugged-in I/O device. The hot plug services software will then begin device discovery.

2.7.2.1.1 Hot-plug messages

There are 7 different hot-plug messages defined in PCI-Express. Every message of this type will be routed to the COP for processing (address_lookup_module with match the bridge header space) if the message is addressing one of the P2P bridge headers in our switch. These are only required if the LEDs are implemented on the downstream I/O device. If the LEDs are directly by the switch (using GPIO), these messages won't be used.

2.7.2.1.2 Attention_Button_Pressed

The Attention Button Pressed register bit must be set to a 1 if our switch receives this message from a downstream port. If our switch detects the attention button was pressed for a given slot (16 I/O pins on our switch, one for each port), our switch will generate an MSI and send it up to the hot plug services software.

2.7.2.1.3 Attention_Indicator_On, Attention_Indicator_Blink, Attention_Indicator_Off

These three messages will be generated by the COP and sent downstream depending on the state of the attention_indicator bit.

2.7.2.1.4 Power_Indicator_On, Power_Indicator_Blink, Power_Indicator_Off

These three messages will be generated by the COP and sent downstream depending on the state of the power_indicator bit.

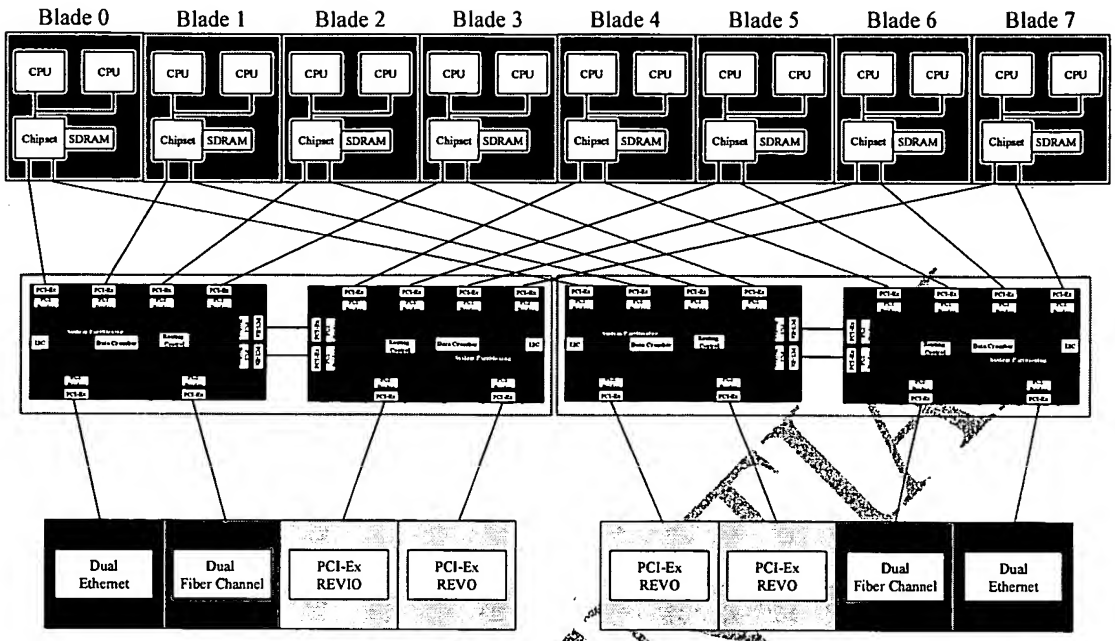


Figure 2.7-1: Server Chassis Example

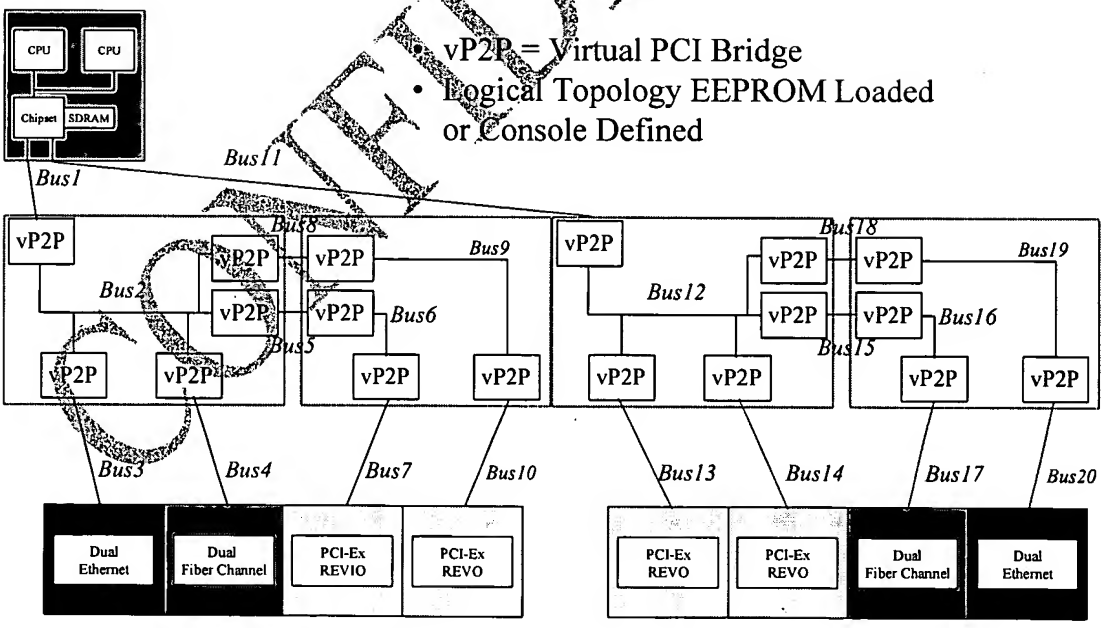


Figure 2.7-2: Logical Bus View